Universidade de Aveiro
2014

Departamento de Eletrónica, Telecomunicações e Informática

Sara Cristina Albuquerque Figueiredo

Desenvolvimento de um Sistema de Diálogo para Interação com Robôs

Development of a Dialog System for Interaction with Robots

Universidade de Aveiro
2014

Departamento de Eletrónica, Telecomunicações e Informática

Sara Cristina Albuquerque Figueiredo

Desenvolvimento de um Sistema de Diálogo para Interação com Robôs

Development of a Dialog System for Interaction with Robots

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor António Joaquim da Silva Teixeira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor José Nuno Panelas Nunes Lau, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Dedico este trabalho aos meus pais, ao meu irmão e aos meus poucos mas grandes amigos, pois sem eles não teria conseguido chegar a esta nova fase da minha vida.

agradecimentos

Gostaria de agradecer o apoio e orientação do Prof. António Teixeira e do Prof. Nuno Lau, que tornaram possível a realização deste trabalho e orientaram-me para uma atitude mais profissional para trabalhos futuros.
Agradeço também o apoio, paciência e compreensão infinita dos meus familiares nos momentos bons e nos momentos de maiores dificuldades.
Não posso deixar de agradecer ao meu grande amigo André pelo seu constante apoio e contribuição na fase de avaliação do projeto, assim como ao Pedro também pela sua contribuição.
Finalmente, à minha grande amiga, Cláudia, que nunca deixou de acreditar em mim.

palavras-chave

Interação humano-robô, robôs de serviço, robô humanóide, sistema de diálogo, gestão de diálogo, comunicação por linguagem natural

resumo

Os robôs de serviço operam no mesmo ambiente dos humanos e executam ações que um humano normalmente executaria. Estes robôs devem ser capazes de operar de forma autónoma em ambientes desconhecidos e dinâmicos, assim como de manobrar em ambientes com várias pessoas e de saberem lidar com elas. Ao respeitarem estes requisitos, conseguirão abordar com sucesso os humanos e cumprir as suas solicitações sempre que estes precisem de assistência em alguma tarefa. A comunicação por linguagem natural, nomeadamente a fala que é a forma mais abrangente de comunicação entre humanos, torna-se relevante na área da Interação humano-robô (IHR). Ao dotar os robôs de serviço com sistemas de voz intuitivos facilita-se a especificação das tarefas a realizar. No entanto, é uma tarefa complicada de se realizar devido aos recursos envolvidos na criação de uma interação suficientemente intuitiva e devido à dificuldade de funcionar em diversos robôs.

O objetivo principal deste trabalho é a definição, implementação e avaliação de um sistema de diálogo que seja de fácil integração em qualquer sistema robótico e que funcione como uma base flexível para qualquer cenário de conversação na língua Portuguesa. Deve obedecer a requisitos base de comunicação intuitiva e natural, nomeadamente a características de conversas entre humanos.

Foi desenvolvido um sistema que funciona como uma base para dar continuidade a trabalho futuro em sistemas de diálogo. O sistema incorpora a arquitetura cliente-servidor onde o cliente é executado no robô e capta o que o utilizador diz. O cliente tira partido de serviços de gestão de diálogo externos ao robô, executados pelo servidor, que processa o áudio obtido, devolvendo uma resposta ao cliente adequada ao contexto do diálogo. O desenvolvimento foi baseado numa análise crítica do estado da arte para se tentar manter fiel ao que já foi feito e de forma a se tomarem as principais decisões durante a implementação.

Mediante a fase de avaliação do sistema, tanto a nível do ponto de vista da interação como do programador, conseguiu-se obter por parte de alguns voluntários que o objetivo principal foi cumprido: foi criada uma base suficientemente flexível para explorar diferentes contextos de conversação, nomeadamente interagir com crianças ou fornecimento de informações em ambiente universitário.

Keywords

Human-robot interaction, service robots, humanoid robot, spoken dialogue systems, dialogue management, natural language communication

abstract

Service robots operate in the same environment as humans and perform actions that a human usually performs. These robots must be able to operate autonomously in unknown and dynamic environments, as well as to maneuver with several people and know how to deal with them. By complying with these requirements, they are able to successfully address humans and fulfill their requests whenever they need assistance in a certain task. Natural language communication, including speech that is the most natural way of communication between humans, becomes relevant in the field of Human-Robot Interaction (HRI). By endowing service robots with intuitive spoken interfaces, the specification of the human required tasks is facilitated. However, this is a complicated task to achieve due to the resources involved in creating a sufficiently intuitive spoken interface and because of the difficulty of deploying it in different robots.

The main objective of this thesis is the definition, implementation and evaluation of a dialogue system that can be easily integrated into any robotic platform and that functions as a flexible base for the creation of any conversational scenario in the Portuguese language. The system must meet the basic requirements for intuitive and natural communications, namely the characteristics of human-human conversations.

A system was developed that functions as a base to give continuity to future work on Spoken Dialog Systems. The system incorporates the client-server architecture, where the client runs on the robot and captures what the user says. The client takes advantage on external dialogue management services. They are executed by the server, which processes the audio obtained, returning an appropriate response given the context of the dialogue. The development was based on a critical analysis of the state of the art in order for the system to be as faithful as possible to what is already done.

Through the evaluation phase of the system, it was managed to obtain by few volunteers the conclusion that the main objective was accomplished: a base system was created that is flexible enough to explore different contexts of conversation, such as interacting with children or providing information on a university environment.

Content

List of Figures

List of Tables

Acronyms

ASR - Automatic Speech Recognition

CARL - Communication, Action, Reasoning and Learning in Robotics

DETI - Departamento de Eletrónica, Telecomunicações e Informática

DM - Dialogue Manager

FSM - Finite State Machine

HCFSM - Hierarchical Concurrent Finite State Machine

HCI - Human-Computer Interaction

HLT - Human Language Technologies

HRI - Human-Robot Interaction

IEETA - Instituto de Engenharia Electrónica e Telemática de Aveiro

IS - Information State

NLG - Natural Language Generation

NLU - Natural Language Understanding

OAA - Open Agent Architecture

RTN - Recursive Transition Network

SDS - Spoken Dialog System

SLU - Spoken Language Understanding

SS - Speech Synthesis

SUS - System Usability Scale

Chapter 1 Introduction

1.1. Motivation

Nowadays there are three major groups of robots, having each one of them sub classifications of their own: industrial, military and service robots. These classifications are built according to the environment of the robots, thus, by their functions and what they are used for. Researchers on industrial robots must focus on their precise motion, dexterous manipulation and so forth, which are requirements for factory environments. On the other hand, researches on service robots should focus on intelligent navigation algorithms for them to adapt into dynamic environments and should focus on natural ways of communication with people, being them by gestures, touch, voice or multimodal, for instance.

The **main focus** of the dissertation is the **service robots** for their relevant necessity to interact with humans to aid them in their daily tasks. It is utterly important for service robots to autonomously operate in a dynamic and unaccustomed environment, as well as to be able to maneuver in environments with multiple people, in order to be able to successfully approach humans and fulfill their requests when assistance is needed. Therefore, depending if the robot is supposed to act as an instrument or as a personal assistant, the aspects of human-robot interaction (HRI) in these types of robotics varies deeply. In order for service robots to perform their roles accordingly, there is the need for intuitive interactions so humans can easily communicate with the robots and state their requests and the robots can understand them and act on those requests.

These challenges and the implementation of an intuitive way of HRI group up the main motivations for developers on HRI for service robots. In addition, there are several concerns about the growth in the percentage of the elderly population in Europe and therefore the European Commission (EC) is requesting more investments in research and development on Information and Communication Technology to help overcome both the problems of lack of human resources and the ageing population [1]. The main focus is to assure more quality of life for the elderly. One example of this is the HERMES (Cognitive care and guidance for active aging) [2]. It resulted in solutions that, for instance, help the elderly cope with memory loss trying to maintain the brain active. Therefore, and considering these needs, by implementing an intuitive way of HRI, **service robots may become one of the main future options for people with special needs**. The adaptation of human-human behavior into robots has given birth to numerous researches in areas such as using robotics for aiding the elderly, for rehabilitation and for other scenarios that involve people with impairments.

HRI has become a growing research field and is also a multidisciplinary field where its contributions come from areas such as engineering, computer science, social sciences and humanities. This research field is thus aimed at improving the interaction between human beings and robots while these perform their roles in their specific environments. The interaction between robots and humans should be as intuitive and efficient as possible, in order to result in effective communications. The research on HRI, especially when the robots cohabit with humans to help them in their daily tasks, also implies many challenges regarding the learning of human relationships and their social behavior and adapting these characteristics into robots. In fact, as human beings, we prefer what is more comfortable and easier to use, unlike having to pick up a manual and study it deeply or even lightly, in order to learn how to interact with a given system.

Furthermore, it is known that speech is the most common way of human-human communication. By creating a system with the ability to manage and mimic human dialogues, especially with communication failure recovery from recognition and synthesis errors, we introduce comfort and ease into human-computer interaction. Let’s imagine talking to a robot or to technology in general as simply as talking to a human: wouldn’t it be both innovating and interesting? And the implications this would get into the impaired people would be remarkable, since the main objectives would be fulfilled: having simplicity and effectiveness in human-robot interactions and offering these people the opportunity of social inclusion and better life quality without any external efforts (by only using their natural way of communication). Yet, there are several counterarguments for whether speech technologies are preferable as the main interaction between technology and humans [1], being one of them the current limitations in existence. For instance, especially in the speech recognition module of these technologies, there are still some limitations on most languages in the recognition process that may lead to the boredom or discomfort of users. This problem and the complexity of such robust systems are difficulties that every researcher on spoken interfaces for robots has to face. Nevertheless, it should not ever be a means for giving up, since achieving such a goal would have a great impact on HRI.

1.2. Context

The Electronic and Telematics Engineering research unit (IEETA) from University of Aveiro has been long interested in the research and development of systems that assist humans in their environments. For instance, the CARL project (Communication, Action, Reasoning and Learning) was developed with the purpose of studying the interrelations and integration of all four dimensions of building an intelligent robot, such as the initials suggest (Communication, perception & Action, Reasoning and Learning) [3]. This robot suffered many improvements from 1999 to

2013 and can communicate with humans via a friendly interface, by voice on both input and output or by text input, in order to assist them by giving information on the whereabouts of teachers at the University and even give directions for visitors. This robot was very popular at University of Aveiro. Another example is a project involving an intelligent wheelchair – IntellWheels [4]- that can be controlled via voice, head movements or even facial expressions and has contributions from IEETA. The project allows for users to choose the most comfortable and suitable input. This project has been awarded five times by national and international entities and has already been tested by patients with cerebral palsy [5], [6].

Other works include the CAMBADA@Home robot [7], under the Living Usability Lab (LUL) project which is a project that aims for the development of solutions in favor of better life quality for people in need [8]. The CAMBADA@Home has participated in several national and international competitions, including the RoboCup world championships, Dutch Open and the annual Portuguese Open Robotics Festival [9]. It was created subsequent to the team past experience in the CAMBADA middle-size robot soccer team.

Although these projects gained a lot of popularity and prestige both nationally and internationally, the HRI does not entirely focus on human-human communication-based interfaces but mostly on a more command-based language which is not always too intuitive for humans. To add, none of the spoken systems available in the current projects has been made that could actually be easily adapted to any robot in future research and development and also to any scenario concerning the robot's environment.

1.3. Objectives

We have seen that a system capable of interacting with a human in a human-like way is an appealing idea. However, considering the context of the research and development in Portugal and in the World, there are few documented works that allow for this type of spoken interfaces. It is relevant to facilitate future work for developers and give continuity to the research; as a result, the main objective of this dissertation is to provide an easy and flexible solution that supports future research and development of spoken interfaces for robots. The contributions include:

- The development of a flexible spoken dialog system, having Portuguese as the main language, taking in consideration the easiness and comfort of its use, as far as usability rules and human communication are concerned;
- This system should be flexible enough to allow it to be easily adapted in different scenarios;

- The system should be as much as possible robot independent to allow it to be easily deployed on any robot, running any operating system.
- The spoken dialog system should allow for characteristics that define an intuitive interaction, such as mixed-initiative dialogues and sense of presence in the communication.

1.4. Document Structure

This chapter attempts to outline the motivation for the work of this Master thesis, as well as provide for some definitions for assistive robotics and the implications of the use of speech in the context of HRI for service robots.

The second chapter, a mix of background information, related work and state-of-the-art, will review the body of research relevant to this work, including a brief explanation of Spoken Dialog Systems and the relevant features that these systems should include in order to achieve an intuitive HRI.

Following the State of The Art chapter, in Chapter 3, it is presented an overview of the system, including diagrams which describe the architecture of the system as well as a detailed explanation of each component and the main decisions and modifications taken throughout the development. This chapter also includes details about the future communication between the implemented dialog system and other robot's functionalities.

The Results chapter presents in detail the evaluations and respective results obtained both for the testing of the system's flexibility and for the system's capacity to perform intuitive interaction with humans.

Finally, the Conclusion wraps up the thesis with a brief overview of the concepts discussed, the main results and an indication of a direction for future work.

Chapter 2 Background and Related Work

2.1 Background

2.1.1. Human-Human Communication & Human-Robot Communication

First of all, while building an effective spoken interface, the developers should be familiar with the basic traits of human-human communication. There are formal descriptions of these traits and one example is the one given by Nickerson [10], also referenced in [11]. The most relevant traits for this thesis are the concepts of:

- ✓ Bi-directionality;
- ✓ Mixed initiative dialogues;
- ✓ Apparentness of who is in control;
- ✓ Sense of presence;
- ✓ Intolerance for silence;
- ✓ Structure;

These concepts are related to very familiar traits that create pleasant dialogues. Considering two people talking with each other, in order to achieve a pleasant conversation: the information should flow from both sides; new information can be added from both participants, at any time (information not related to the current context of conversation); there must be the feeling that the other participant is actually present in the dialogue and following the flow of the dialogue; among other traits. In dialog systems in general, having the ability to embrace these traits and also confirm what was said or giving the sense of understanding is fairly relevant for the user's perspective. However, this enclosure has costs, especially in complexity, because achieving such a spoken interface still has limitations and the HRI field is a very complex one. In terms of conversational robots, these should have the ability to understand clear and complete commands as well as resolve ambiguities and complement missing information. One relevant aspect is the handling of low confidence inputs, when the robot tries to confirm if what was understood is correct. For the sake of long-term human-robot relationships these should also give the sensation that the robot understands the user's feelings, by evaluating the registered history of the dialogues as well as its current context and by showing the appropriate empathy to the user. Other important aspects are stated on [1]. Furthermore, robots should allow for humans to interrupt the current action when needed as well as change the current domain of conversation, for more robust communications.

2.1.2. Spoken Dialog Systems (SDS)

While a dialog system is mainly a program capable of performing communication with humans, a spoken dialog system (SDS) is mainly a computer system which is capable of performing communication with humans through voice [12]. Therefore, it is a system that should be able to recognize the human voice, get the meaning of what was said and give an adequate synthesized answer, building natural human-computer conversations. Examples of applications with spoken dialog systems include the Siri app [13] and Microsoft Cortana [14] as intelligent personal assistants, Let's Go! [15] for bus scheduling information, Jupiter [16] for worldwide weather forecast information over the telephone, call center applications for automatic handling of customers and, of course, most of the recently built robots which are endowed with this technology, such as the WITAS UAV [17], a small autonomous helicopter.

In order for a spoken dialog system to work properly, it depends on a wide range of Human Language Technologies (HLT), being them: automatic speech recognition (ASR), natural language understanding (NLU) or, using another name considered more adequate by some authors, Spoken Language Understanding (SLU), dialog management (DM), natural language generation (NLG) and speech synthesis (SS). These components and their connection can be seen in Figure 1.

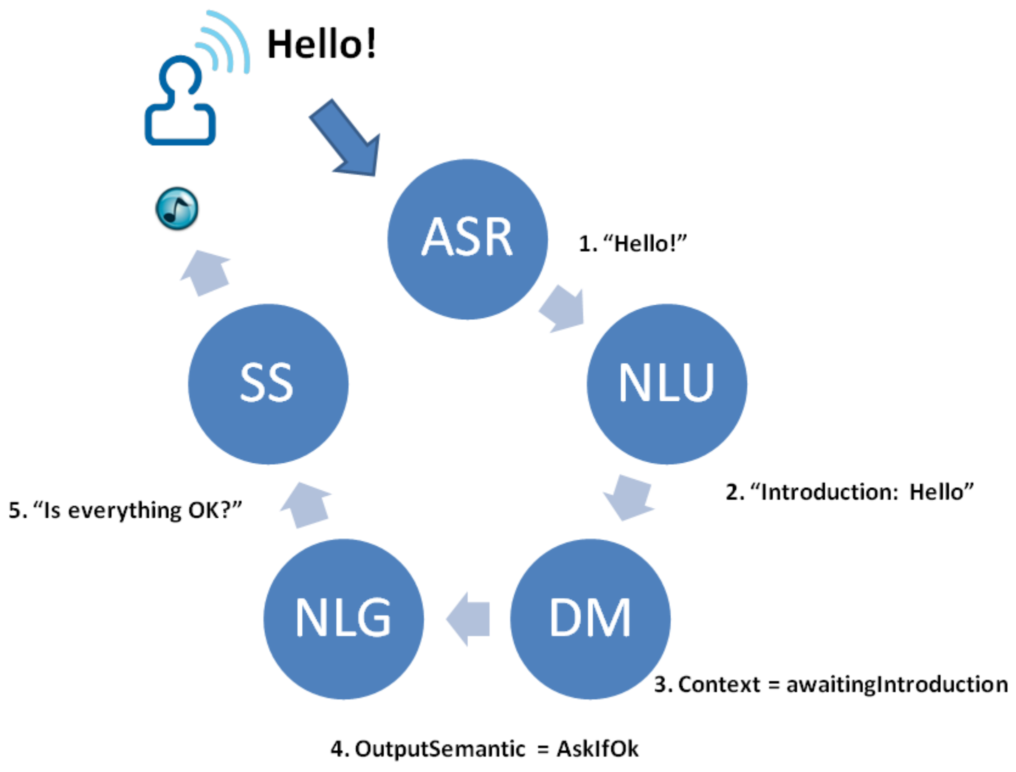


Figure 1: Typical Architecture of a Spoken Dialog System

The ASR is responsible for the receiving of an audio input and translating it into text. Then, the NLU adds semantic to the recognized text, either by hand-written grammars, either by more complex processes. This semantic is useful for the DM to analyze what to do next, given the user input. The DM is the component which deals with the state of the communication and its secret lies beyond the kind of representation of *dialogue state* that is used which defines the information needed/given at a certain time. The state of the dialogue is updated according to user utterances understanding, dialogue history and all other relevant information concerning the dialogue and the context of the application. According to [18], also referenced in [11], there are three types of DM: **Finite State Systems**, **Frame-Based Systems** and **Advanced Systems** which, for now, the Information State (IS) is the most relevant. For example, if the dialogue state is defined by a Finite State Machine (FSM) then for each user utterance with the according semantic, it should decide the next state on the dialogue according to the event that was generated by this utterance, on the FSM. If, on the other hand, the DM is of frame-based type then the current state is defined of whether the important frames have been properly filled in order for the system to give an answer to the user. The Information State type is based on a structure that contains the dialogue state variables, somehow like an object in an object-oriented paradigm. These variables are updated according to pre-defined rules that can involve other variables or user utterances. The more sophisticated the DM the more flexible and manageable the dialogues. Finally, the DM sends a message to the SS component, either a semantic value that is a match in a template file that belongs to the NLG grammar, either directly the full text to be synthesized.

The purpose of this dissertation does not include an extensive explanation of each component and available technologies. Therefore, more explanations will be given when needed throughout the next chapters.

2.2 Related Work on HRI for Service Robots

Service robots should be able to understand human requests to perform tasks and provide humans with some necessary information. As far as long-term HRI is concerned, the communication should be intuitive, natural and sometimes, depending on the robot's focus, entertaining. If the voice is the most natural way of human-human communication, why not incorporate the robot with a sophisticated Spoken Dialog System which can easily understand voice commands and generate appropriate answers, as well as serve as a companion for the human?

An overview of the research and development is described on Section 2.2.1. The systems mentioned and other related systems are analyzed in detail concerning the implementation and capacities of the different modules that are part of a SDS. The conclusions obtained will be explained in Section 2.2.2.

2.2.1. Overview

There is not plenty of research in terms of robots interacting with people. One of the most famous robots nowadays is the Asimo humanoid robot [19], for its complexity. There is also the Nao by Aldebaran [20] with its small and funny appearance, which has been incredibly used in helping children with autism with the Ask Nao solution [21]. Another example is the most recent robot from PAL Robotics, REEM-C [22]. Research in human-robot interaction per voice has been focusing a wide range of areas. For example, work has been done that uses natural language to overcome difficulties in localization and navigation problems [23]. On the other hand, the Nao robot has been used not only to help children with special needs but also as a test bed for investigating child-robot interaction by developing imitation of arm movements and quiz games [24]. Additionally, the Robovie-IV [25] is a robot that interacts with people daily in an office environment to help them whenever they need assistance. Unfortunately, there is a lack of information of how the Spoken Dialog Systems were built for the robots and also a lack of functionalities which could fairly improve their flexibility and empathy towards humans. An in-depth analysis is summarized in the form of a table. This table has been divided to facilitate the comprehension of its contents. Table 1 refers to the general information on the robot and the implemented DM. Table 2 focuses on the modules providing information to the DM. Table 3 focuses on the NLG and SS modules and Multimodality features. Table 4 focuses on the available languages and other features such as empathic/emotional interaction, user experience adaptation and flexibility. Finally, Table 5 describes the SDS in terms of the evaluation methods.

First of all, it is important to label some words used to fill the tables. Said this, the reader can understand the following from the table's entries:

- ✓ **N/a (not available)**: means that no information was available;
- ✓ **No**: means that the current data was implicitly found as nonexistent in the dialog system;
- ✓ **Yes**: means that the current data was implicitly found as existent in the dialog system;

The chosen references focus on service robots, although some of them, such as the PeopleBot robot, were developed in order to study a specific field: supporting long-term human-robot relationships by managing Facebook information and creating a form of "shared memory". Some robots from the Robocup@Home competition were also chosen as this is a huge international competition and a good reference for comparison. In addition, there are other works that mention the use of speech but no information regarding the SDS was found ([26], [27], [28]). Firstly, it will be presented detailed information regarding the chosen robots for the tables. This will then be followed by a critical analysis that combines the information collected for the considered robots.

The robot Carl, which has been created at Universidade of Aveiro, with the purpose of studying the integration of all four dimensions of building an intelligent robot, has an information-state (IS) dialog manager [11]. The state is updated according to the robot's Open Agent Architecture (OAA) agents, being them the NLU, GTI (Graphical and Touch Interface) and Navigation agents. The dialog system allows for domain switching, low confidence, interruption handling and mixed-initiative dialogues. The vocabulary size of the grammars is of small/medium size and the NLU component uses semantic networks to determine the semantic relationships and speech act present in the user utterances. The NLG is based on Multimodal Functional Unification Grammar, which is a development and debugging tool for natural language generation, to create the according sentence to be synthesized. There is no empathic or emotional interaction but the system adapts to user experience by storing information given by the user. In order to evaluate the system, interaction, usability and functional tests were made and the results show that, generally, the system worked well for the majority of the tests. The results can be better seen in Table 5.

The Reem-B is one of the PAL Robotics robots. It was designed to communicate with people and perform sophisticated tasks while cohabiting with humans. According to the work done in [29], the system uses a hierarchical and asynchronous FSM as the Dialogue Manager, where the states can be sequential or concurrent. Their state transitions can be triggered by external events such as TCP-IP messages, vocal events or even by the person's face. There was no data available whether the system is capable of interruption and low confidence handling, as well as if domain switching is allowed. There is no empathic or emotional interaction and there are no details about the evaluation methods for the speech module.

The IntellWheels project [30] aims at providing an intuitive and flexible solution for wheelchairs that helps people with special mobility needs. The implementation focuses on a multimodal interface that combines modalities such as speech, touch, facial expressions and head gestures. Each user input takes the form of a sequence that is the preferred user inputs/action association. This input can be the combination of the different input modalities. For instance, it can have a combination of simple speech command such as “go” and a head tilt to the right, and this specific sequence will perform a specific action on the wheelchair. The speech input management is hence simple. It takes advantage of the IBM Via Voice capabilities for the recognition process. The dialogue management is built in a multimodal manager performed by a multimodal control interface, which analyses each user input combination. A very important conclusion to take note concerning this project is that it performs user adaptation. While presenting a multimodal interface, this project provides options to the patients and lets them choose the most comfortable and suitable input.

The HERMES robot was designed and tested with the purpose of raising awareness for research on integration and dependability, and for long-term human-robot interaction experiments. It was used as a museum guide for six months ([31], [32], [33]). The only information available concerning the structure of the SDS was that it supports interruptions by the users, user initiative dialogues and that the ASR component makes use of a simple, small and fixed grammar that supports mostly imperative sentences that have a relatively simple structure. The NLU component is based on a command interpreter that collects lexical, syntactical and semantical analysis from the recognized text in the format of, for instance, “GO([location])”. This command interpreter, after receiving the text string given by the ASR, uses a set of delimiters to create a sequence of words and numbers that are then given a type by the lexical analyzer. A type is anything from verbs, locations, prepositions, objects, and so on. After that, the syntactical analyzer proceeds with the identification of the sentence structure by using a list of prototype command sentences that define what the robot can understand. If the comparison between the list of types and the list of command sentences is successful, then the semantic analyzer will provide missing words (such as “it”) from the robot’s situated knowledge in order to complete the command. In terms of evaluation, this robot was tested on TV studios, trade fairs and in a museum for six months. The results determine that the robot was capable to interact dependably with people and their common living environment.

The PeopleBot robot was another robot used for long-term human-robot relationships research [34]. It is one of the two most interesting entries considering user experience adaptation, allowing for Facebook information retrieval and sharing to perform dialogues. There is no information on the type of DM. However, domain switching is allowed, allows for robot initiative dialogues and adapts to user experience by analyzing a “shared memory” related to Facebook news and database contents. Therefore, it can create utterances such as “I saw Peter yesterday!” and “it’s good to see you again, Sara!”. Considering the evaluation of the SDS, no speech data was available.

The Team Description Papers of the Robocup@Home 2014 were considered by the author as relevant for the state of the art analysis. The Robocup@Home is the largest international annual competition that encourages teams from all over the world to participate in many challenges that may prove their innovative contributions in the area of autonomous service robots [35]. The main scenarios are home environment and, therefore, one of the main research fields is HRI and Cooperation. Therefore, and also because all Team Description Papers were similar in layout and contents, four robots were chosen to be included on the table: the KeJia [36], ToBI [37], Golem-II+ [38] and Lea [39]. For the latter, a mail was sent to request a paper that fully explained the implemented spoken dialogue system but no answer was received. There is a lack of detailed information for all these robots. Available information for the KeJia robot shows that the DM uses a FSM to represent the dialogue state and that it supports robot initiative dialogues. Considering the Golem-II+ robot, the only dialogue management information obtained is that it is based on recursive transition networks, a graph theoretical schematic used to represent the rules of a context-free grammar, where nodes represent world situations and edges represent expectations and action pairs. The available information also states that for the ASR component they used a hand-crafted corpus for each task, where the ASR is able to switch from one to the other. The NLU functions according to stored regular expressions and their meanings and further matching them to the orthographic transcriptions that are similar to the expectations of the system. A deep semantic parser based on the Grammatical Framework formalism [40] is also available. No information regarding the speech evaluation process and results could be found for any of the robots, because only the general scoring for each team could be retrieved.

Regarding the ASIMO robot, the works of [41] include a very detailed information on the SDS. The DM is a combination of frame-based management and hierarchical planning, made by the use of what they call experts, each with its own information state. These experts can process request understanding, information providing, physical action planning and information obtaining action. They are implemented as objects in object-oriented paradigm. In terms of interaction, the work states that the SDS supports domain switching, low confidence handling as well as interruption handling. The ASR component uses network grammars as recognition language models, with a vocabulary size of about 400 words. On the other hand, the NLU gives semantic representation based on utterance patterns and keyword lists. The dialogues support no empathic or emotional interaction, but are of mixed-initiative type. Furthermore, no information was obtained regarding the evaluation process.

As previously said, the Nao robot was used as a test bed for investigating child-robot interaction by developing imitation of arm movements and quiz games ([24], [42]). In this work, a DM of information state type was implemented, where the action-selection mechanism is a Non-deterministic FSM. Additional information regarding the SDS indicates that the ASR focuses on N-best list output and adaptation techniques and has a large vocabulary size (no specific data available), and the NLU gives semantic according to a hand-written grammar based on the Multimodal Combinatory Categorial Grammar framework (CCG). For more information on CCG the user is invited to read [43]. Concerning the NLG, the text to be synthesized is determined via canned text sent from the DM or via a deep-generation approach with utterance content planning on the basis of a communicative goal specified by the DM. In terms of empathic or emotional interaction, by the time the paper was written only non-verbal feedback was considered and they were testing the Mary TTS capabilities. More recent work states that it was indeed integrated [42], allowing for prosody and voice quality control. Two implemented approaches are used: one focus on the direct decreasing of the speech rate and a raise on the pitch contour on some words in the NLG sentences; another focus on the DM's decision to render the output with emotional characteristics such as "sad" or "happy" emotions, which are possible by the increasing or decreasing of speech rate and pitch contour. Considering the evaluation process, technical evaluations of the intended functionality and Wizard-of-Oz experiments were prepared. However, the system showed not to be mature enough for end-to-end usability evaluation due to its lack of robustness on the recognition and interpretation of inputs.

2.2.2. Detailed Information

Starting the detailed analysis with the robot's main purpose and environment and the implemented Dialogue Manager, Table 1 presents the type of implemented DM and other aspects relevant to the dialogue management such as domain switching capabilities, low confidence recognition handling and dialogue/sentence interruption handling.

Robot	Description	Dialog Manager			
		Type	Interruption Handling	Domain Switching	Low Confidence Handling
Carl [11]	**Developed at UA to study integration of 4 dimensions of building an intelligent robot**	**Information-state**	**Yes**	**Yes**	**Yes**
Reem-B [29]	**Designed to cohabit with humans**	**Hierarchical and asynchronous FSM**	**N/a**	**N/a**	**N/a**

IntellWheels [6], [30]	**Developed to aid people with physical injuries**	**N/a**	**Yes**	**N/a**	**N/a**
HERMES [31], [32],[33]	**Designed and tested for dependability**	**N/a**	**Yes**	**N/a**	**Yes**
PeopleBot [34]	**Helping managing Facebook information**	**N/a**	**N/a**	**No**	**N/a**
KeJia [36]	**For use at unprepared environments**	**FSM**	**N/a**	**N/a**	**N/a**
Jijo-2 [44]	**Operates in office environment: guides visitors, delivers messages, etc.**	**State transition network. Current state is a state in a finite state in a automation network**	**No**	**Yes**	**Yes**
ToBI [37]	**For use at unprepared domestic environments**	**N/a**	**N/a**	**N/a**	**N/a**
Nao [24], [42]	**Test bed for investigating child-robot interaction in the ALIZ-E project**	**IS. The action-selection mechanism is a Non-deterministic FSM**	**N/a**	**N/a**	**N/a**
ASIMO [41]	**Service humanoid robot for tasks as schedule management and weather info**	**Combination of frame-based and hierarchical planning**	**Yes**	**Yes. Based on hand-written heuristic rules**	**Yes**
Golem-II+ [38]	**For use at unprepared domestic environments**	**Based on recursive Transition Networks**	**N/a**	**N/a**	**N/a**

Table 1: State of the Art - Robots' Description and Dialogue Management

According to the analyzed references, the most common type of dialog management involves Finite State Machines. There are some as well, as it was previously mentioned on Section 2.2.1., that are based on an Information State dialogue management where the information state is determined by either other components such as the robot's agents, either by the NLU or even external TCP-IP messages, depending on the robot's purpose and environment. In spite of the majority of the entries being filled with "not available" (n/a) information, the majority of them show that the DMs are domain switching and perform low confidence and interruptions handling.

Table 2 presents information regarding the Speech Recognition such as the common adopted systems for the recognition process, the type of grammars used and the recognition rate during the testing phases. Information on the NLU module is also presented, considering the process of semantic assigning.

Robot	**Speech Recognition and NLU**				
	ASR System	**ASR Grammar**		**ASR Recognition Rate**	**NLU**
		Description	**Vocabulary Size**		
Carl [11]	**Nuance 8.0**	**Based on Lexical Functional Grammar**	**Small/medium**	**N/a**	**Picks up the syntactic analysis of the recognized input and, based on a semantic network, determines the semantic relationships and speech acts**

Reem-B [29]	**N/a**	**N/a**	**N/a**	**N/a**	**N/a**
IntellWheels [6], [30]	**IBM Via Voice**	**Very Simple Command-based sentences**	**N/a**	**N/a**	**Via the multimodal manager. Gives a definition that the user input contains a speech component part and defines the speech command based on pre-defined simple commands.**
HERMES [31],[32], [33]	**N/a**	**Simple and fixed grammar that supports imperative sentences that have a simple structure**	**Small**	**N/a**	**Based on a Command Interpreter made of a Parser and Lexical, Syntactical and Semantical Analysis components**
PeopleBot [34]	**Sphinx 4**	**N/a**	**Small**	**No**	**N/a**
KeJia [36]	**SAPI**	**N/a**	**N/a**	**N/a**	**Syntactic parsing (Stanford parser) and semantic representation using λ-calculus**
Jijo-2 [44]	**Ninja, and hidden Markov model-based system**	**Phoneme grammars, a word dictionary and a grammar created beforehand.**	**N/a**	**86% with standard microphone. 47% for Omni directional microphone**	**Simple word dictionary, with task-dependent semantic equivalences**
ToBI [37]	**CMU Sphinx**	**N/a**	**N/a**	**N/a**	**N/a**
Nao [24], [42]	**CMU Sphinx**	**N-best list output and adaptation techniques.**	**Large (not specified, but thought around hundreds)**	**N/a**	**Hand-written grammar based on Multimodal Combinatory Categorial Grammar framework**
ASIMO [41]	**Julian**	**Network grammars as recognition language models**	**400 words**	**N/a**	**Based on utterance patterns and keyword lists (Speechbuilder based)**
Golem-II+ [38]	**PocketSphinx coupled with WSJ acoustic models**	**Hand-crafted corpus for each task. The ASR can switch from one to the other**	**N/a**	**N/a**	**Storing regular expressions and their meanings and further matching to the orthographic transcriptions OR using deep semantic parser based on the GF formalism**

Table 2: State of the Art - ASR and NLU components

Beginning with the ASR module, the CMU Sphinx is the most used system for speech recognition. The ASR grammar is mostly defined by hand-written grammars, consisting of simple and fixed corpus. Half of the Table 2 entries for the vocabulary size could not be filled as it was not mentioned in the documents, but the majority of the available ones show that the ASR modules are built with a small corpus, and it is believed that by small they mean around hundreds of words. Only one entry mentions the recognition rate obtained in the recognition process, which is the one that belongs to the Jijo-2 robot. The process of assigning semantic to the recognized inputs, which belongs to the NLU component, is mostly based on lists/dictionaries with semantic equivalences. Thus, a matching is done between the syntactical analysis of the recognized input and the lists or dictionaries available in the systems. This matching is done via the help of domain construction tools such as the SpeechBuilder for the ASIMO robot and the Multimodal Combinatory Categorial Grammar framework for the Nao robot, which as was previously explained, is a development and debugging tool for natural language generation.

Continuing the detailed analysis with the methods for generation of natural language and the Speech Synthesis component, Table 3 summarizes how the chosen references perform the generation of natural language (after the verification of the dialogue context) and the chosen systems for the synthesis of the NLG's generated content.

Robot	NLG	Speech Synthesis		Multimodal Interaction ?	
		System	Available Genders	Input	Output
Carl [11]	**Multimodal Functional Unification Grammar**	**Festival**	**Male**	**Touch, Text and Speech**	**Text, Speech and Facial Expressions**
Reem-B [29]	**N/a**	**N/a**	**N/a**	**Speech**	**Speech and Physical Actions**
IntellWheels [6], [30]	**No.**	**No system**	**No system**	**Speech, Touch, Head gestures and head movements**	**Graphical display and wheelchair movement**
HERMES [31],[32], [33]	**N/a**	**N/a**	**N/a**	**Text, Speech and Touch**	**Text, Speech and Physical Actions**
PeopleBot [34]	**N/a**	**Cepstral TTS**	**N/a**	**No**	**No**
KeJia [36]	**N/a**	**N/a**	**N/a**	**Speech Only**	**Speech and Physical Actions**
Jijo-2 [44]	**Set of reply templates with a context-based slot-filling method**	**N/a**	**N/a**	**Speech**	**Speech and Physical Actions**
ToBI [37]	**N/a**	**Mary TTS**	**N/a**	**Speech, Gestures (not specific which), Touch**	**Speech and Physical Actions**
Nao [24], [42]	**Canned text sent from the DM OR a deep-generation approach with utterance content planning on the basis of a communicative goal specified by the DM**	**Acapela TTS. Mary TTS**	**N/a**	**Speech and Gestures (left hand up and down, right hand up and down & combinations)**	**Speech, Body Language for Expressing Emotions such as anger, sadness, fear, happiness**
ASIMO [41]	**N/a**	**FineVoice by NTT-IT**	**N/a**	**Speech**	**Speech and Physical Actions**
Golem-II+ [38]	**N/a**	**Festival TTS**	**N/a**	**Speech**	**Speech,**

Physical Actions

Table 3: State of the Art - NLG, SS and Multimodality

Taking into account the NLG, one reference points out the use of templates with slot-filling and another one mentions the use of canned text sent from the DM. Another one also mentions the use of Multimodal Functional Unification Grammar which is a version of Functional Unification Grammars, a formalism widely used for text generation, applied for multimodal situations. Apart from this, FESTIVAL and Mary TTS are the most used systems for the synthesis of the generated text given by the NLG and there was no information available concerning the available genders, apart from the Carl robot which uses a Male gender. Further analysis of the Table 3 indicates that half the references mention the use of multimodality in the input. The majority of them only have speech and physical actions as output. Some of them allow for the users to perform gestures, such as few hand movements (up and down) or even a combination of them.

Table 4 presents information on the chosen languages for the interaction. Other interaction details are also present such as: the type of initiative that the system allows; if the use of empathy or emotions throughout the dialogues is considered; and if the SDS adapts somehow to the users experience. A relevant characteristic for the system's functionality, its flexibility for other domains, is considered.

Robot	Languages	Initiative			Other		
		Robot	User	Mixed	Empathic/Emotional Interaction	Adapts to User	Flexible for Other Domains
Carl [11]	**English**			**X**	**No**	**Yes. Stores info given by the user**	**Yes**
Reem-B [29]	**N/a**		**N/a**		**No**	**N/a**	**N/a**
IntellWheels [6], [30]	**N/a**		**X**		**No**	**Yes. Provides options to users and lets them choose the most comfortable and suitable input.**	**Yes.**
HERMES [31], [32], [33]	**English, French, German**		**X**		**No**	**N/a**	**No**
PeopleBot [34]	**English**	**X**			**No**	**Uses Facebook news based on a "shared memory"**	**N/a**
KeJia [36]	**English**		**X**		**N/a**	**N/a**	**N/a**
Jijo-2 [44]	**Japanese**		**X**		**N/a**	**N/a**	**No**
ToBI [37]	**N/a**		**N/a**		**N/a**	**N/a**	**Yes**
Nao [24], [42]	**Italian**		**N/a**		**Non-verbal feedback. Mary TTS allows for changes of speech rate and pitch to create**	**Has a user model to store user name and previous game scores, but no**	**N/a**

				“sad”, “happy” or “stress” situations	additional info of how this is used	
ASIMO [41]	**Japanese**		**X**	**No**	**No**	**Yes**
Golem-II+ [38]	**Spanish**	**N/a**		**N/a**	**N/a**	**Yes**

Table 4: State of the Art - Languages, Dialogue Initiative and Other Aspects of Interaction

Only one of the spoken dialog systems is multilingual, the one that was used on HERMES robot, and none of them has been developed in the Portuguese language. Most of them were built for the English or Japanese languages. Moreover, the majority of the dialogues are of user initiative, being them to ask the robot to carry out a certain task and wait for the robot to actually perform it.

The only entry that shows consideration for empathy in the dialogues, as well as some emotional features, is the one regarding the Nao robot. It states the use of facial expressions or body language that gives the user some feeling that the robot is actually cooperating and understanding the context of the dialogues. In addition, they recently integrated the Mary TTS system that allows for some changes of prosodic members in the sentences. To conclude, the majority of the table entries show no empathic/emotional interaction in the dialogues. Table 4 also confirms that there is a lack of adaptation for the user experience throughout time and the dialogues. The IntellWheels project is the only reference that mentions different user choices giving different outputs in the dialogue (multimodal, in this case) management. Another interesting concept is the use of Facebook data to create a form of “shared memory” that may alter the course of the dialogues. Others, such as the ones used by robot Carl and Nao, store some user data but on the latter there is no data explaining how this is actually used in the dialogue management. The majority of the research and development show that the systems are flexible for other tasks/domains, in spite of the lack of information.

To conclude the analysis, Table 5 presents information on the evaluation phase of the developed systems.

Robot	Evaluation	Results
Carl [11]	**Interaction, usability and functional evaluation (10 participants)**	**It was easy to understand the robot but not so easy to make the users understandable. The rhythm of interaction was appropriate and the robot acted as predicted for the majority of participants. The DM was capable of managing the interaction in an efficient way, providing knowledge for both sides**
Reem-B [29]	**N/a for speech**	**N/a**
IntellWheels [30], [6]	**Some evaluations performed in simulated room with obstacles. Evaluation of different inputs (combined or not). Evaluations were made also to compare between the system command options for the**	**For the voice command testing, seven commands (front, back, left, right, turn, go and stop) were defined for the voice input to control the wheelchair. When using voice control, it was concluded that it is preferable for open areas without obstacles because of the delay in the response of the speech recognition software. Considering the therapists**

	user and the ones given from therapists (11 participants with cerebral palsy participated in these evaluations)	and system options comparison, it was concluded that the system results are very similar to the ones recommended by therapists. The generated command language had even better evaluation.
HERMES [31]–[33]	TV studios, trade fairs and tested for more than six months in Heinz Nixdorf MuseumsForum (an undefined yet large number of participants)	Showed to interact dependably with people and their common living environment. In dialogues and other interactions it appears intelligent, cooperative and friendly.
PeopleBot [34]	Module-level, system-level and task-centered evaluations (no speech data). Task-center was carried out with five people, each one of it four times	
KeJia [36]	N/a	
Jijo-2 [44]	N/a	
ToBI [37]	N/a	
Nao [24], [42]	Technical evaluation of the intended functionality. Wizard-of-Oz experiments (no participant information)	The event-based control mechanism and the interfaces between components work as intended. There are certain limitations of the robot platform for future designs to take into account. Both the recognition and interpretation of inputs (speech and gestures) are not robust enough for untrained users. Thus, not mature enough for end-to-end usability evaluation
ASIMO [41]	N/a	
Golem-II+ [38]	N/a	

Table 5: State of the Art - Evaluation

Table 5 suggests that there is also a lack of evaluation information. One with some detail is the one performed with Carl. Interaction, usability and functional evaluations were carried out, with ten undergraduate and postgraduate students from Universidade de Aveiro [11]. It can be observed from the collected data that most systems could be understandable by the users but not so easy to make the users understandable, since the recognition process still has some limitations. For instance, the evaluations carried out for the IntellWheels project concluded that the speech input was considered better for open areas due to the delay in the speech recognition. The works done with the HERMES robot show that the system was able to interact dependably with people and their common living environment, mostly due to its easiness in maintainability [31]. It was tested both by completely novice users (when in trade fairs, television studios, for instance) and also in long-term experiments (museums and offices). Despite the first table entry, the one from Carl robot, there is a relevant lack of information regarding the usability of the spoken interfaces.

2.2.3. Final Comments

The overall analysis of the research and development of Spoken Dialog Systems in service robots indicates that it is relevant to give continuity and support for more research. There is a serious lack of information on the developed projects concerning their spoken interfaces. Besides, on some of the most detailed references, they do not consider the implementation of functionalities that could fairly improve the SDS overall flexibility or empathy towards humans. The most complete section of information in the investigated papers is frequently related to the hardware description of the robots, giving very little or almost no reference to the software used for the spoken interaction. Consequently, it was hard to fill the tables with complete SDS information. Nonetheless, a few conclusions were taken that helped into guiding this Dissertation.

The State of the Art analysis left the perspective that the recently built robots are all endowed with speech recognition and synthesis capabilities. If the latter is not present, other convenient types of output are considered. Although the used technologies could not always be detailed, it can be concluded that at least for simple words/sentences the robots are capable of performing simple dialogues with humans. Only one reference mentions the use of both non-verbal and verbal empathic and emotional feedback for the humans. This is an important characteristic to contribute for longer term human-robot relationships, since service robots may become one of the future options to help people with special needs. Considering the users adaptation to the interaction, it is not enough to save general user data into databases, such as the name and the last time he/she interacted. The IntellWheels project considers the user's choice between a set of defined input modalities, contributing to a more comfortable interaction. The system advises as well the best options for driving the wheelchair, where the best set of input sequences are included, as well as its association with the available commands. Since this was the only reference presenting an example of interaction modes alternation, it would be equally relevant for longer term human-robot relationships to create other solutions that consider the user adaptation. Another perspective that the analysis left is that no spoken interface was implemented for the Portuguese language, not even with the robot Carl which was developed at University of Aveiro. The most important aspect to consider, also bearing in mind the objective of this dissertation, is that most of the systems are flexible for the adaptation into other dialogue domains and even have domain switching capabilities.

This analysis was done in order to guide the dissertation. The result should be in a similar level as the state of the art, and above in some characteristics, to contribute with a simple and interesting solution that could give continuity to the research and development of SDS. Since most of the investigated papers present flexible systems, the resulting system needed to be flexible and of easy adaptation into any scenario. It was decided that the system should make use of empathic/emotional interaction, as well as some strategies to improve the interaction through time, for the sake of longer human-robot relationships. In addition, it was important that the system would be developed in the Portuguese language.

Chapter 3: Spoken Dialog System Architecture and Implementation

This chapter explains the architecture of the developed Spoken Dialog System (SDS), taking into consideration the roles of the users and the robot throughout the system components. The implementation of each component and the main improvements made throughout the development process are detailed.

3.1. Architecture

First of all, the current implementation is based on a previous **client-server architecture** that was developed during a research scholarship of another student from Universidade de Aveiro [45]. Its purpose focused on the development of a simple spoken solution that could be easily deployed in a robot running the ROS (Robot Operating System) on top of a Linux kernel or with similar systems. The Robot Operating System is the most famous collection of software frameworks to build robot applications [46].

The developed architecture focuses, thus, on the client-server architecture. The client is intended to run locally in the robot and the server to run remotely. Usually, a robot does not have enough capacity to deal with huge amounts of processing on its own and a real issue continues to be the battery life of the robots. They can sometimes require external machines to perform some of the operations, depending on the robot's environment. The developed solution contributes to the decrease of the processing overload inside the robot, since the only component which is necessary to run inside the robot is the client and it is the simplest component of the solution. Therefore, this architecture simplifies the robots' processing by giving more autonomy concerning one of their major issues nowadays: the battery life. The client-server solution also allows for multiple robots to connect to the server, where each client can run inside any robot with any operating system (OS), being agnostic to the machine's OS that runs the server. Moreover, taking the dialogue management in consideration, by having the server running on a different machine with different capacity, the management can be easily maintained. The typical scenario is the one illustrated in Figure 2.

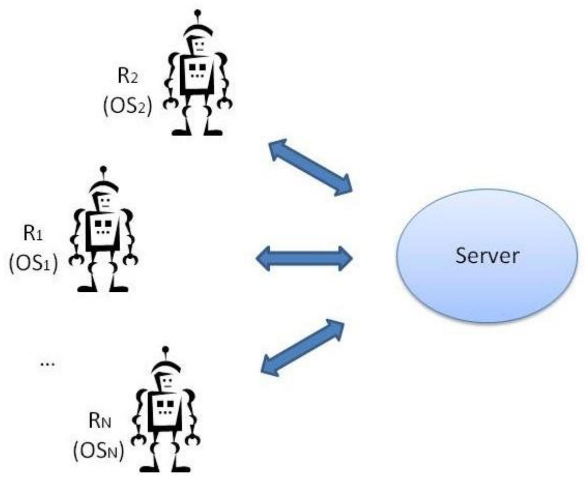


Figure 2: Client-Server Architecture - Robot Independence Overview

The architecture considers, then, different **clients running inside robots with different Operating Systems** and one **server**, typically running on a different machine, **which performs the operations of a Spoken Dialogue System**. The role of each robot can be seen in Figure 3. The robot has the client running inside it, as well as other robot applications. The **client** has two main components which are represented as **filled circles** in the figure. One is responsible for the reception of speech events from humans (any spoken words/sentences) and for sending the audio streams to the server. Another client's component is responsible for the reception of a dialogue response, coming from the server after analyzing the dialogue context and the last speech event, and playing it back for humans to listen. **Other** robot's **applications**, represented by the **non-filled circle**, can communicate with the DM module and vice-versa.

To explain better, the client detects the default recording device and continuously monitors for audio streams coming from the user(s) (the humans). After receiving an audio stream, the robot sends it to the remotely localized server that will process this information as a typical Spoken Dialog System does. If the robot is provided with other applications that should communicate with the dialogue management or vice-versa, those applications can contribute to the dialogue context as well. For instance, if the robot encounters a relevant object on its way, such as a cup that the user was looking for, with the help of its vision and mapping modules, it makes sense to send an event to the dialogue that may trigger special sentences. One example of a sentence could be one that informs the user that the robot has entered, for instance, the kitchen and that it encountered the relevant object and prompts the user what it should do next, stopping other previous dialogues.

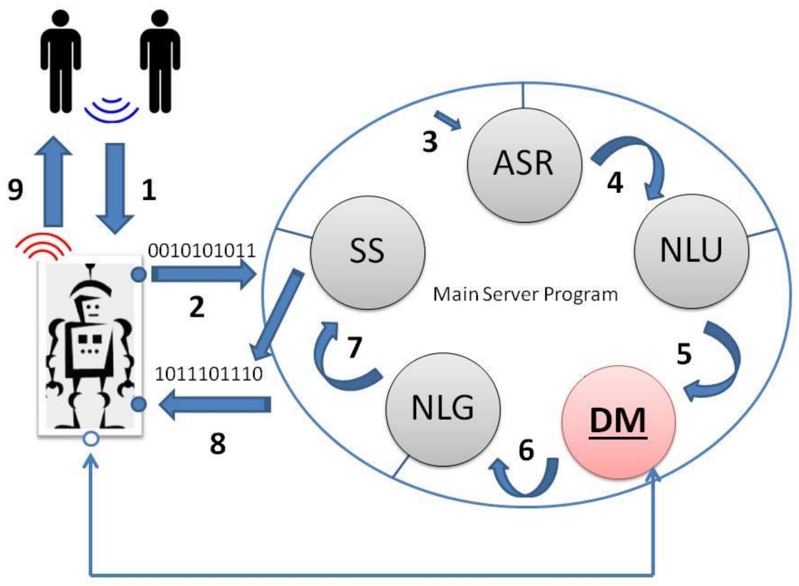
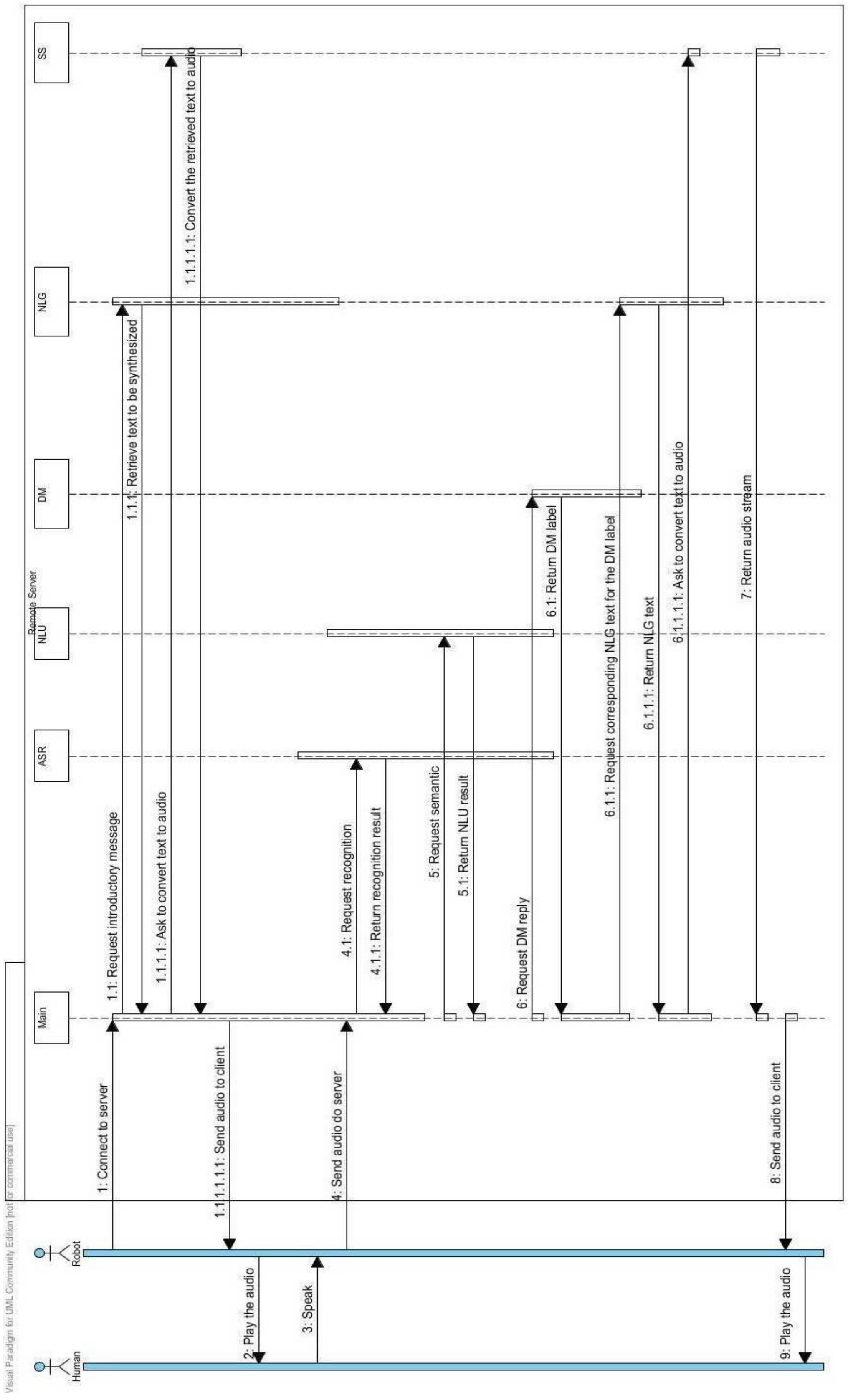


Figure 3: Client-Server Architecture – The Role of Each Client and the Server

The server, thus, continuously listens for recognition requests (audio streams that need to be converted into text) from the connected client(s) (the robots). After receiving an audio stream, the server first tries convert it into text with the aid of the **ASR** (Speech Recognition) module. Secondly, the **NLU** (Natural Language Understanding) module gives semantic to this text. The **Dialog Manager** then analyses the context of the dialogue and returns an appropriate label, the NLG label, for the **NLG** (Natural Language Generation) module to retrieve the reply to be sent to the **SS**. The SS module synthesizes this reply into audio and the server sends it back to the robot. At that moment, all the robot needs to do is use its default playback audio device in order to play this audio stream for the users to listen.

Although the complete adaptation of the SDS into a robotic platform is not included in the objectives of this dissertation, the future deployment of the developed SDS will have to consider as well the integration of other robot's components such as the ones that handle gestures, for instance, or even mobile and vision issues to create more complete dialogues. Therefore, **the DM will not only manage user utterances in order to give a proper reply to humans**, but **also** consider other contexts involving **robots' mobility, vision and other processing** that may be relevant for the dialogues.

The developed architecture can be better understood in the sequence diagram in Figure 4. This sequence diagram generally describes all steps beginning with the connection of the client to the server until the procedures taken whenever humans say something to the robot.



Remote Server
Main
ASR
NLU
DM
NLG
SS
Robot
Human
1: Connect to server
1.1: Request introductory message
1.1.1: Retrieve text to be synthesized
1.1.1.1: Ask to convert text to audio
1.1.1.1.1: Convert the retrieved text to audio
1.1.1.1.1.1: Send audio to client
2: Play the audio
3: Speak
4: Send audio do server
4.1: Request recognition
4.1.1: Return recognition result
5: Request semantic
5.1: Return NLU result
6: Request DM reply
6.1: Return DM label
6.1.1: Request corresponding NLG text for the DM label
6.1.1.1: Return NLG text
6.1.1.1.1: Ask to convert text to audio
7: Return audio stream
8: Send audio to client
9: Play the audio
Visual Paradigm for UML Community Edition [not for commercial use]

Figure 4: Client-Server Architecture's General Sequence Diagram

After a client connects to the server, getting an ID representing its session and a dialogue configuration, then dialogues can start to take place and the server, after successfully connecting the client, requests the NLG for an introductory message and synthesizes it into another audio stream to be played by the robot. As soon as the robot plays this introductory message, it waits for human utterances to continue the dialogue. The server can also send other audio streams to the client apart from dialogue responses for the received utterances. For instance, there are **system events**, such as timeouts or other relevant information, which are important for the user perspective. This information is not represented on the sequence diagram.

The base which makes this whole processing of human utterances possible, as well as these system events, is described in the following chapters.

3.2. Implementation

Both the roles of the client and the server were already explained in the previous chapter: the client is intended to run on the robot, receive audio input from users, send it to the server and play the received response given by the server; the server is intended to run remotely and, after receiving an audio stream, performs actions according to a typical SDS, sending an appropriate dialogue response back to the client. Although the objectives of this dissertation do not include a complete adaptation of the SDS into a robotic platform, the server's DM module is supposed to be ready for the reception of inputs from other robot's components that contribute for the Dialogue Management. This chapter will detail how both the client and the server can perform these actions.

3.2.1. The Client

The client was developed in Java$^{\text{TM}}$. While monitoring the audio signals, the speech events that are captured through the received audio streams are based on a threshold of the sound amplitude that indicates when someone speaks. When the threshold is reached, it allows for the client to start recording and finish the recording when the audio amplitude drops to the threshold again. The task of sending the whole stream to the server, remotely localized, is done via the use of a recompiled version of the Java-WebSocket client [47]. Taking this into consideration, and for correctly running the client, at least the first two arguments of the ones detailed below must be defined, in the following order:

1. The host for the WebSocket URI;
2. The desired threshold (a number between 0 and 1);
3. An audio file for testing purposes if some is available. This audio is stored in a previously defined directory on the machine running the server. This was implemented because the audio files recorded by the client are not in a standard format to be reproduced by a typical player, such as VLC. Taking this into consideration, this solution is important for debugging the architecture regarding the recording and recognition processes.

The server will eventually send an appropriate response in the format of another audio stream that will be played by the robot's default audio output device, detected by the client.

3.2.2. The Server

The server, in order to perform actions as a typical SDS, is much more complex than the client. It consists of a set of sub modules, as Figure 2, Figure 3 and Figure 4 illustrate, where the processing is sequential. The main server program, as well as its sub modules, was implemented in C# in order to facilitate the development within Visual Studio^{TM}. This was also chosen because the recognizer and the synthesizer are from Microsoft^{TM}, turning the development easier within Visual Studio. Each sub module will be explained in the following sections, but first the connection of the client will be taken in consideration.

The job of the server is to continuously listen for recognition requests (the audio streams to be processed). In order to do this, a client-server connection and its identification are performed to correctly manage the dialogues of each session. In order to allow for client-server communication, SuperWebSockets [48] was used on the server side, which is a .NET implementation of WebSocket server and that allows for events to be handled. There are four types of events that can occur:

- `NewSessionConnected(WebSocketSession);`
- `NewDataReceived(WebSocketSession, Byte[]);`
- `NewMessageReceived(WebSocketSession, String);`
- `SessionClosed(WebSocketSession, CloseReason);`

Figure 5 demonstrates an activity diagram of what really happens when the event `NewSessionConnected` is generated. When a client successfully connects to the server it is given a `Session_ID`, the ID of the WebSocket session, and a session configuration. This configuration is the global information that defines the SDS. Therefore, it is the chosen language, the grammars, the speech recognizer instance, the dialog manager instance and the natural language generator instance. Because of the ID of the session and its configuration, the system can handle multiple clients. When this configuration is properly set, then the client and the server are ready to perform dialogues.

It is important to mention that each session has a counter for the number of seconds that have passed since the server sent the last synthesized NLG sentence to be played on the robot. These seconds are relevant to trigger a special message from the server when the user(s) have not talked for a certain amount of time. This time is defined according to the length of the last NLG message plus a number which can be changed (this value depends on what is considered an ideal waiting number until a new user utterance is received). When this time is reached, the main server program calls a method from the DM to get a special NLG label and to update the dialogue context, if necessary. This method will be better explained in Section 3.2.2.3.

It is also important to note that, after a new client is connected, the session information (session ID and the session configuration results) as well as the SDS procedures whenever a new user utterance is received on the server side (recognition results, semantic assigning results, and so on) is **saved in a log file**, **locally stored in the machine running the server**. Therefore, the session configuration for each client, any warnings, errors and each SDS module results are stored in this file for debugging purposes and also to keep a history of the dialogues.

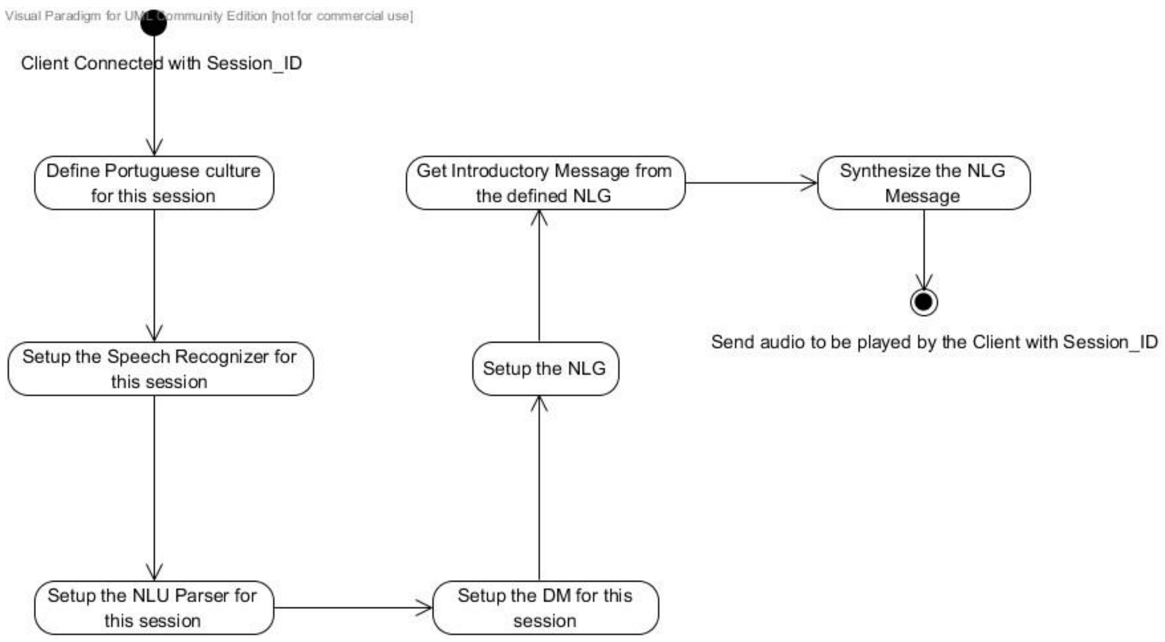

Visual Paradigm for UML Community Edition [not for commercial use]
Client Connected with Session_ID
Define Portuguese culture for this session
Setup the Speech Recognizer for this session
Setup the NLU Parser for this session
Setup the DM for this session
Setup the NLG
Get Introductory Message from the defined NLG
Synthesize the NLG Message
Send audio to be played by the Client with Session_ID

Figure 5: `NewSessionConnected` Event Handling

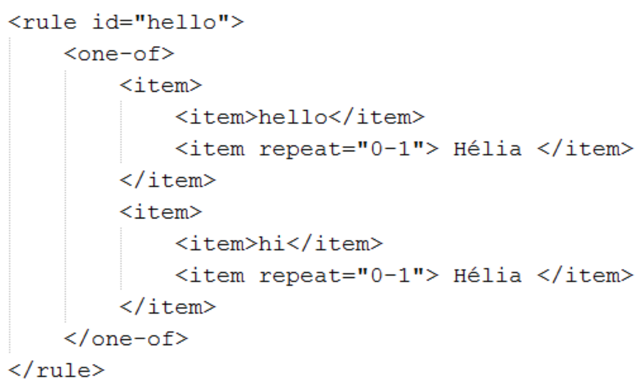
The `NewMessageReceived` event is generated whenever the client sends a text message indicating whether the robot is currently "speaking" the last synthesized NLG sentence. This event is triggered twice: when the robot begins to play the received audio stream from the server and when the robot stops playing it. Thus, this event changes the value of a global variable that indicates "`true`" or "`false`" concerning the playback of the last audio stream. This is relevant in the sense that the main server program knows when the robot is playing an audio message and, thus, can perform additional decisions. The client does not stop recording and sending speech events and the client should be kept as simple as possible. As a result, the fact that the client does not stop recording at any time in its life cycle means that speech events which are considered as "garbage" (not relevant) for the context of the conversation were sent to the server. In addition, the further processing of these non relevant speech events could result in inconsistencies within the dialogue management. Therefore, whenever the client is playing a synthesized NLG sentence, no further processing will be done to the next received speech events. However, it makes sense the inclusion of a **stop command in the user perspective**. All human-robot interfaces should be provided with an option that allows for the user to stop the current action being performed by the robot. In dialogues this also makes sense, therefore the stop command was implemented so the user can make the robot stop speaking whenever it is adequate. In order to detect if a stop command was indeed sent, all **audio streams** must **pass through the speech recognition and the natural language understanding modules** in order to decide if the processing should continue to the DM or not. To conclude, **no dialogue management operation will be called whenever the robot is playing an audio message, discarding everything in between (only passing by the ASR and NLU modules to verify what was said).**

The `SessionClosed`, as the name suggests, is generated whenever a connection with a client is closed. Both the `Session_ID` of the previous connection and the reason for the connection closure are printed in the Console for debugging purposes.

Another event that is handled on the server side is the `NewDataReceived`. This event is generated whenever a client sends a byte array containing the audio stream to be processed. This is the event that toggles the main SDS processing.

3.2.2.1. Speech Recognition

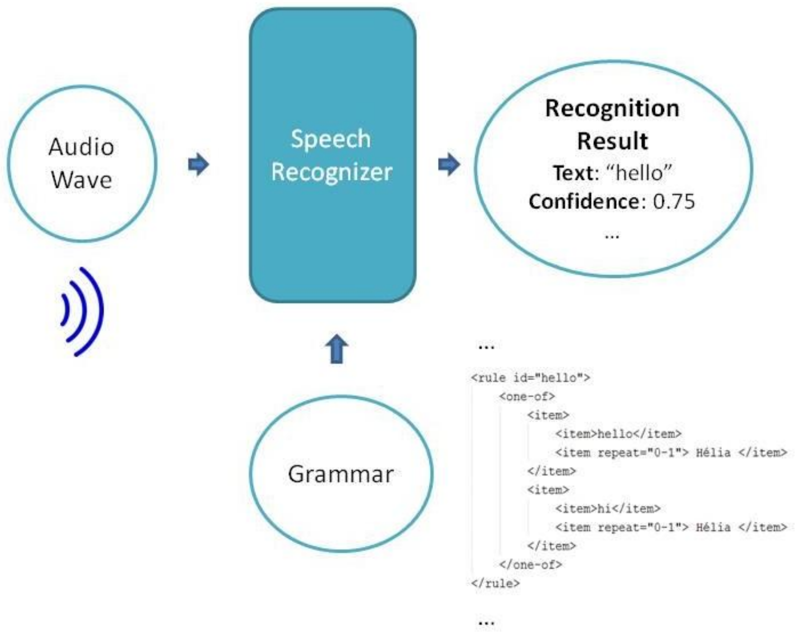
This is the module responsible for the recognition of text within an audio stream. Thus, given an audio wave, a grammar and the speech recognizer, the result, if something can actually be recognized, is the corresponding text of what was said in the audio. To contextualize, after the client-server connection, if the `NewDataReceived` event is called then it means that the client detected a speech event and sent it to the server. The first thing to do is recognize what was said in this audio data. During the configuration setting when performing a client-server connection, a recognizer is set and it is given a grammar and a culture to set the speech domain. The speech recognizer that was used was the one from Microsoft^{TM}. The Microsoft Speech Platform version 11 was the version used [49], and it provides a set of complete tools that can provide applications with spoken interfaces that give effective and natural ways of human-computer interaction (HCI). Moreover, the chosen Runtime Language for speech recognition was the Portuguese language. The definition of the speech recognizer is made by giving it a pt-PT culture and also a file path, defined by a global path variable, for the grammar file. This grammar file follows the Speech Recognition Grammar Specification [50] and is a hand-written grammar. An example of a user greeting definition in the grammar file can be seen in Figure 6. The figure illustrates that the grammar format follows a XML format.



```
<rule id="hello">
    <one-of>
        <item>
            <item>hello</item>
            <item repeat="0-1"> Hélia </item>
        </item>
        <item>
            <item>hi</item>
            <item repeat="0-1"> Hélia </item>
        </item>
    </one-of>
</rule>
```

Figure 6: Example of ASR Grammar Definition

This grammar works as one of the inputs for a speech recognizer. It informs the recognizer about the collections of words and sentence patterns where these words may occur (which are typical responses from the users of the respective SDS) and that the recognizer should listen for. For example, by using the example in Figure 6, the recognizer knows it should listen for the words "**hello**" ("olá" in PT), "**hi**" ("oi") and "**Hélia**". The patterns these words can follow are: "**hello Hélia**" ("olá Hélia"), "**hello**" ("*olá*"), "**hi Hélia**" ("*oi Hélia*") or "**hi**" ("*oi*"). Figure 7 shows an example of the recognition process involving an audio wave that represents, for instance, a "hello" speech event. By matching the audio stream against the grammar it produces the recognition result "hello" with other relevant information, such as the confidence of the recognition process and a set of alternative grammar words, if available.



```
...
<rule id="hello">
    <one-of>
        <item>
            <item>hello</item>
            <item repeat="0-1"> Hélia </item>
        </item>
        <item>
            <item>hi</item>
            <item repeat="0-1"> Hélia </item>
        </item>
    </one-of>
</rule>
...
```

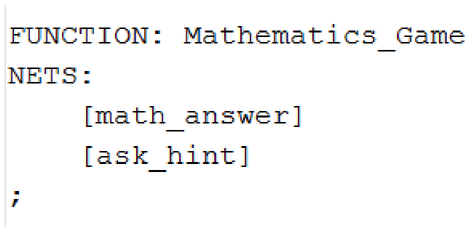
Figure 7: Recognition Process - The ASR inputs and outputs

Consequently, after trying to recognize something from the audio, the result is stored in a class defined by: the recognized text, the confidence of the recognition, any alternatives for the recognition (if available) and other relevant information. If something was recognized, then the text is sent to the NLU for future parsing. If nothing was recognized, then a special method from the DM is called which may update the dialogue state. This will be described in Section 3.2.2.3.

3.2.2.2. Natural Language Understanding

This is the module responsible for giving semantic to the recognized text. To contextualize, when in this module, the main server program knows the recognized text and needs to obtain its semantic to send to the DM even more complete information. During the configuration setting when performing a client-server connection, a NLU parser is set and it is also given a grammar. The NLU is implemented by using Phoenix parser [51], created at CMU, a part of the Olympus/Ravenclaw framework for implementing SDS. The Phoenix parser runs in a separate process and parses an input sentence in order to extract the semantic of that recognized sentence. Considering the Phoenix terminology, this semantic is represented as a **Phoenix frame** with a set of **Phoenix slots** that represent relevant information for that frame. For example, by having an input sentence to be parsed, Phoenix will return one or more Phoenix frames, where each frame has a set of slots that specify the word strings that can fill each slot. Hence, the grammar in Phoenix is actually a set of sub-grammars, each of which attempts to identify a fragment of the input (for example, there is the sub-grammar for confirmation, denial, and so on). Phoenix parses an input sentence and returns results from all these sub-grammars that successfully matched some part of the input. Therefore, Phoenix uses a type of Recursive Transition Networks (RTN). It is important to note that these grammars were hand-written.

For a game scenario where both mathematics queries (from the robot) and answers (from the user) were possible, a Phoenix frame could be `Mathematics_Game`, defined according to Figure 8, and its main slots could be *`[math_answer]`* and *`[ask_hint]`*, also defined according to Figure 8 and detailed on Table 6.



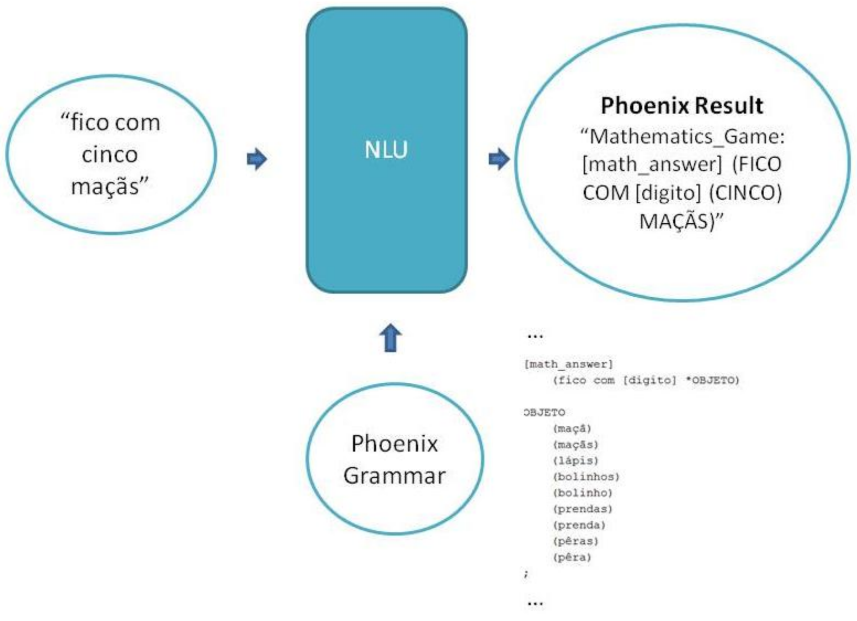
```
FUNCTION: Mathematics_Game
NETS:
     [math_answer]
     [ask_hint]
;
```

Figure 8: Definition of a Phoenix Frame

Defined Grammar	***Translation***
[math_answer]	*[math_answer]*
*(fico com [digito] *OBJETO)*	*(I stay with [digit] *OBJECT)*
*(a resposta é [digito] *OBJETO)*	*(the answer is [digit] *OBJECT)*
OBJETO	*OBJECT*
(maçã)	*(apple)*
(maçãs)	*(apples)*
(lápis)	*(pencil)*
(…)	*(…)*
;	*;*
[ask_hint]	*[ask_hint]*
(ajuda)	*(help)*
(ajuda-me)	*(help me)*
(preciso de ajuda)	*(I need help)*
(dá-me uma pista)	*(give me a hint)*
*(não sei *a *resposta)*	*(I don't know *the *answer)*
;	*;*

Table 6: Example of Phoenix Slot Definition

There is the main slot *[math_answer]* with the sub slots *OBJECT* and *[digit]*, and also the main slot *[ask_hint]*. The word strings that can fill these sub slots are defined right under the main sentences, such as *OBJECT*, by being part of the current main slot declaration (until the semicolon), or in other region of the grammar, such as *[digit]*, when these slots can be used as global. For more information on how to write semantic grammars in Phoenix [51] is recommended. Considering this example, Figure 9 illustrates the process of semantic giving. Although the figure presents an example in the Portuguese language, the translation was already considered and can be seen on Table 6.


"fico com cinco maçãs"
NLU
Phoenix Result
"Mathematics_Game: [math_answer] (FICO COM [digito] (CINCO) MAÇÃS)"
Phoenix Grammar

```
...
[math_answer]
    (fico com [digito] *OBJETO)

OBJETO
    (maçã)
    (maçãs)
    (lápis)
    (bolinhos)
    (bolinho)
    (prendas)
    (prenda)
    (pêras)
    (pêra)
;
...
```

Figure 9: Semantic Giving Process - The NLU inputs and outputs

On the Related Work Chapter, in Section 2.2.2, it was concluded that the process of giving semantic to recognized inputs is mostly based on lists/dictionaries with semantic equivalences with the help of some domain construction tools. Phoenix gives semantic equivalences by dividing the input sentence into sub grammars in order to retrieve the main semantic. In addition, by doing this it can **remove noise from the input** sentence by only matching what matters via these sub-grammars, parsing only the non-noisy parts of the sentences and ignoring the rest, which is an advantage for speech recognition. Some feedback has already been obtained [52] that conclude that Phoenix is an efficient and robust parser for speech recognition.

The extracted frame and frame slot(s), as well as the recognized text, **are** subsequently **sent to the DM** so the manager can analyze what to do according to the dialogue state, by calling its main method. The NLU and DM implementation, especially given the DM's first FSM approach, **considers only the Phoenix slots as relevant for the update of the dialogue state,** because different Phoenix slots can trigger different transitions, depending on the scenario. Therefore, only three types of frame definitions were made: confirmation, help and scenario specific sentences.

3.2.2.3. A First Version of Dialogue Management

The DM is responsible for managing the dialogue. It has a representation of the *dialogue state* and therefore uses the recognized text and the corresponding semantic in order to decide what to do to this *dialogue state*. However, some other system events can trigger a change in the dialogue state, such as when the user does not say something for a certain time as previously discussed. The state of the dialogue is updated according to speech or system events and the DM returns an adequate reply/label to the next phase of the SDS: the NLG. It was already mentioned that when fully deploying a SDS into a robotic platform, the DM can also take in consideration other robot components to update the dialogue state. This can be achieved by allowing these components to send other relevant events directly to the DM, where they are handled by its methods and that may change the DM's *dialogue state*.

As previously mentioned, there are three main types of DM: FSM, Frame-Based and Advanced (IS). After the Related Work analysis on SDS for service robots, it was concluded that the most common type of dialogue management used in the examined documents involves FSM. Consequently, for the first implementation of the DM the chosen type was a FSM based. According to each user speech event (or system event), the DM decided which event to generate in the FSM to update the current state. This was implemented using State Chart XML (SCXML), the "State Chart extensible Markup Language", recommended by the W3C (World Wide Web Consortium) [53]. This language provides a complete set of rules for the definition of event-based state machines and is also used by the W3C Voice Extensible Markup Language (VoiceXML) to define functionality [54]. SCXML also allows for the implementation of HCFSM (Hierarchical Concurrent Finite State Machines). There are many SCXML implementations, including SCION.NET which was the adopted solution. SCION.NET allows for the C#-SCXML communication, by providing lightweight CLR bindings to the SCION SCXML/Statecharts library. SCION consists of an implementation of the W3C SCXML in Java and uses SCXML files as its configuration method. Those SCXML files use JavaScript to provide a faster way to develop new scenarios. Figure 10 presents the implementation of the DM in a general way. The DM was implemented in both C# and SCXML, where the C# side handles the event generation and other relevant verifications and the SCXML side holds the state machine definition.

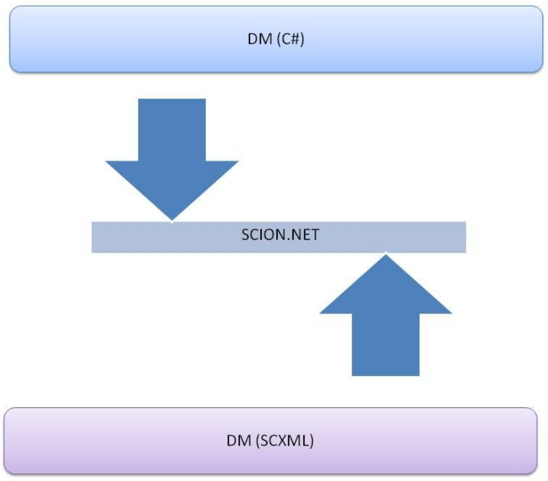


Figure 10: Overview of the DM's Implementation

The implemented FSM follows the structure presented in Figure 11. A hierarchical structure was chosen where the state "`parent`" defines all scenario specific states and their transitions. Outside of the "`parent`" state is the definition of the special states: the "`stopped_hearing`" and the "`low_confidence`" state. These states are special because they are not scenario specific and **each scenario specific state can transit to them**. The first is triggered when the user does not speak for several seconds (defined according to the size of the previous sentence given by the NLG to the SS) and the second when what the user said is recognized with low confidence. Their transitions are defined outside of the main body of the state chart (outside of the "`parent`" state) to maintain a more structured SCXML code. Also, another aspect to take into consideration, in the "`low_confidence`" state, is **the event which was about to be generated if the recognized text is in fact correct**. Because the FSM leaves its main hierarchy of states and waits for a new recognized text and semantic, there is the need to save the previously recognized text and semantic in an auxiliary variable so that if a confirmation is received, then everything works correctly.

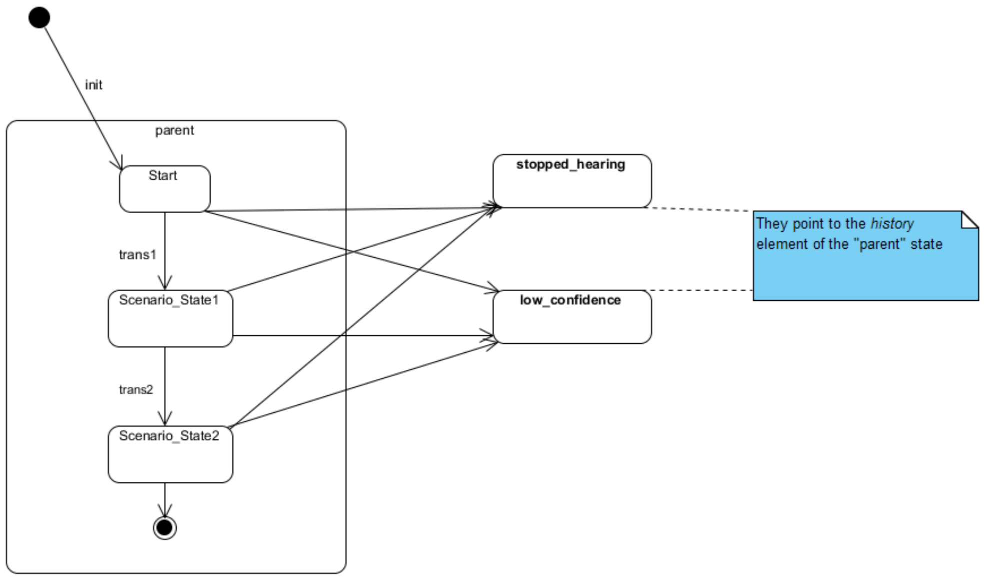
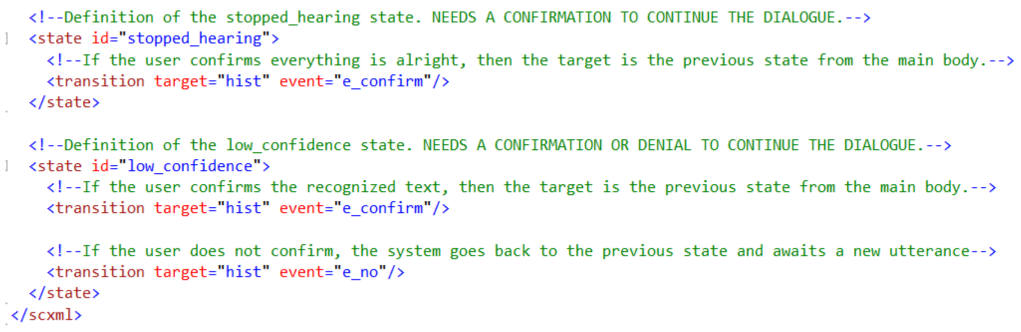


Figure 11: DM'S Finite State Machine Structure

The events that trigger transitions back to the main hierarchy of scenario specific states depend on the special state. The "`stopped_hearing`" state awaits a confirmation from the user. The previous synthesized NLG message, when the transition to this state took effect, prompts the user if everything is alright, so the DM will wait for a confirmation in order to continue the scenario specific dialogue. The "`low_confidence`" state awaits a confirmation or a negation since the previous synthesized NLG message prompts the user if what the system recognized was correct, because it was recognized with low confidence and needs to be confirmed. In the same way as the "`stopped_hearing`" state, the DM will wait for one of these inputs (confirmation or a denial) in order to continue the scenario specific dialogue. The implementation could be improved by allowing other possibilities for these states to transit back to the main hierarchy of states, but this was not implemented in the developed version. To implement these special cases, the **variable *`history`*** of the SCXML specification was used in order **for the system to know where to return in the main body** of the FSM, as soon as these situations were resolved. An example of the syntax used for the special states definition is presented in Figure 12: Special States Definition on the SCXML Note that `hist` is a variable of type *history* on the SCXML specification.



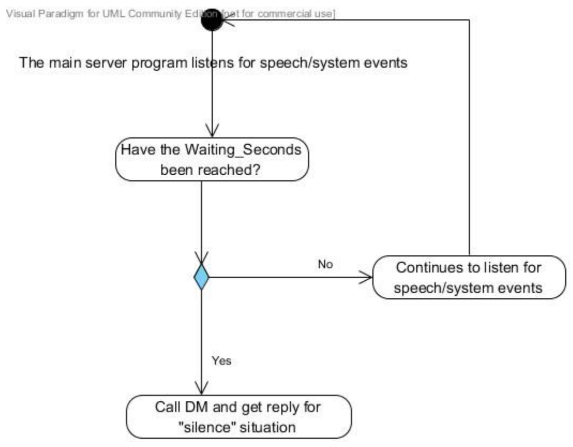
```
<!--Definition of the stopped_hearing state. NEEDS A CONFIRMATION TO CONTINUE THE DIALOGUE.-->
<state id="stopped_hearing">
  <!--If the user confirms everything is alright, then the target is the previous state from the main body.-->
  <transition target="hist" event="e_confirm"/>
</state>

<!--Definition of the low_confidence state. NEEDS A CONFIRMATION OR DENIAL TO CONTINUE THE DIALOGUE.-->
<state id="low_confidence">
  <!--If the user confirms the recognized text, then the target is the previous state from the main body.-->
  <transition target="hist" event="e_confirm"/>

  <!--If the user does not confirm, the system goes back to the previous state and awaits a new utterance-->
  <transition target="hist" event="e_no"/>
</state>
</scxml>
```

Figure 12: Special States Definition on the SCXML

To continue the explanation of the dialogue management processing, it has been explained so far that the DM, after analyzing the *dialogue state* and the inputs that may trigger updates, is responsible for the returning of a NLG label that the NLG module will analyze in order to get a sentence to be synthesized. First of all, two activity diagrams are needed to illustrate what is known so far about the DM's methods. Figure 14 illustrates an activity diagram that represents what happens on the server since the receiving of the audio stream from the client until the synthesizing of the NLG's sentence. Figure 13 illustrates the silence situation already explained, related to the "`stopped_hearing`" state, when the system has not received any audio stream for a certain time. By first analyzing Figure 13 that illustrates a very important system event for the dialogue management, it can be seen that when the user does not speak for a certain time, the main server program calls the DM's method that returns a reply for a "stopped hearing" situation. This method generates a stopped hearing event into the FSM that makes it change its state to one of the special states in the interaction, the "`stopped_hearing`" state. This method also returns a label for the NLG to retrieve the corresponding message to be synthesized. The user is informed that the system has not heard of him for quite some time and is questioned if everything is alright.



Visual Paradigm for UML Community Edition [not for commercial use]
The main server program listens for speech/system events
Have the Waiting_Seconds been reached?
No
Continues to listen for speech/system events
Yes
Call DM and get reply for "silence" situation

Figure 13: Activity Diagram Illustrating a Silence Situation

By now analyzing Figure 14 which presents the general processing of the main server program, it can be seen that when nothing could be recognized from the audio stream, the main server program calls a DM's method that returns a reply for a "no recognition" situation. This method simply returns a "`repeat`" label for the NLG to retrieve the corresponding repeat message to be synthesized. In this specific situation, the user is asked to repeat what he meant.

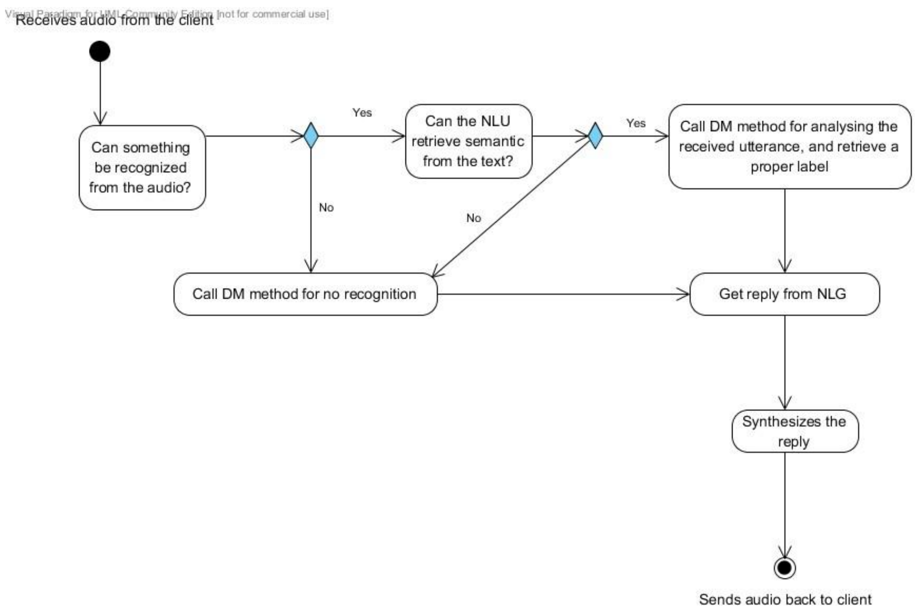
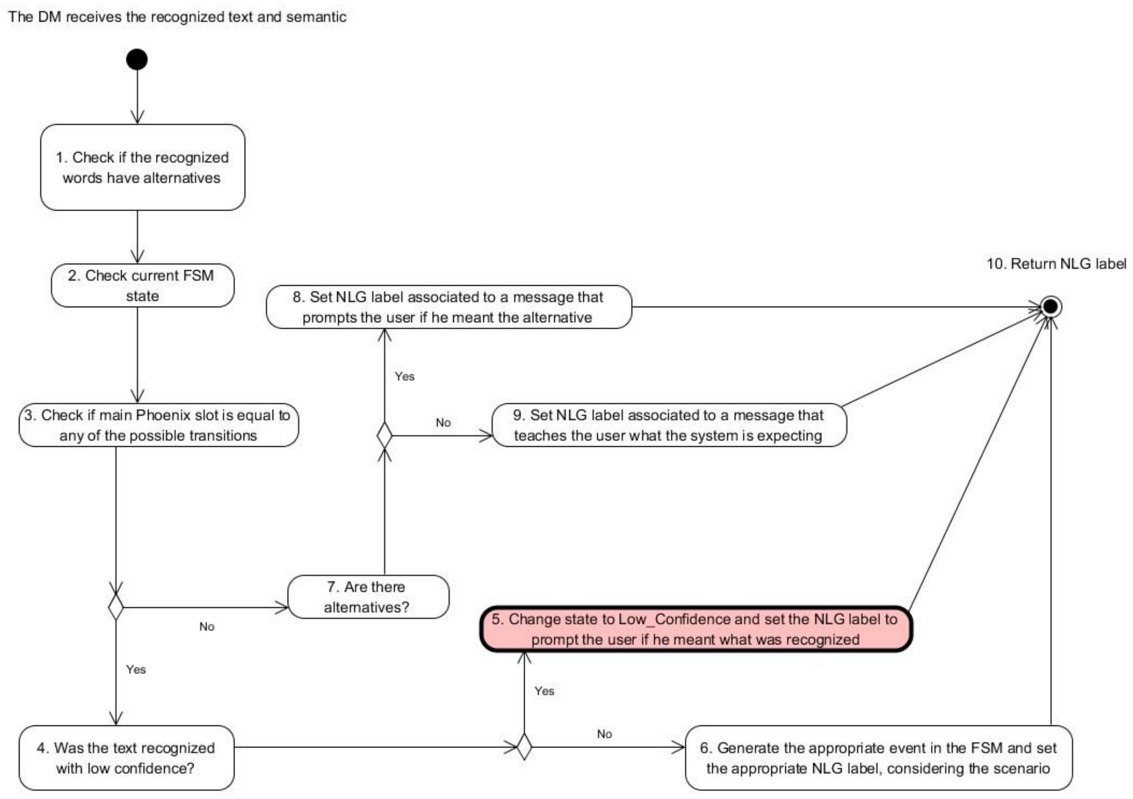

Visual Paradigm for UML Community Edition [not for commercial use]
Receives audio from the client
Can something be recognized from the audio?
Yes
No
Can the NLU retrieve semantic from the text?
Yes
No
Call DM method for analysing the received utterance, and retrieve a proper label
Call DM method for no recognition
Get reply from NLG
Synthesizes the reply
Sends audio back to client

Figure 14: Main Server Program Processing since the Reception of an Audio Stream from Client

When the ASR and the NLU successfully retrieve a recognized text and semantic, respectively, then the main method of the DM is called. This method follows a specific sequence as seen in Figure 15. To recall, **the DM's FSM approach considers only the Phoenix slots (and not the Phoenix frame) as relevant for the update of the FSM state**. Therefore, even if a Phoenix frame called "`Confirmation`" is defined with the slots `[confirm_yes]` and `[im_ready]` (both represent types of confirmations in the user perspective), only the specific slots are relevant to trigger transitions on the FSM because **different slots from the same frame can trigger different transitions, depending on the scenario**. Throughout this section, the reader can sometimes encounter the word "Phoenix slot", "NLU slot" or sometimes the word "NLU semantic", but they represent the same information.

First, it is checked if the recognized words have alternatives. By alternatives it is meant words that are easily confused on the recognition process, such as the confusion between "**`sim`**" ("`yes`") and "**`cinco`**" ("`five`") in the Portuguese language. If the recognized text contains one or more of these words, a global variable is set that can be used later if nothing was recognized considering the current *dialogue state* (see cases 7. and 8. in Figure 15). As a result, **if the**

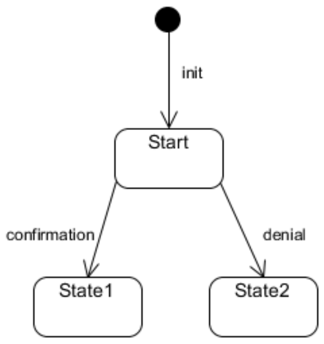
recognized text is outside of the current dialogue context, and the recognized words have alternatives, then **the server first checks if the user meant something else because there is a high probability that the alternative word(s) makes sense for the current dialogue context**.



The DM receives the recognized text and semantic
1. Check if the recognized words have alternatives
2. Check current FSM state
3. Check if main Phoenix slot is equal to any of the possible transitions
No
Yes
4. Was the text recognized with low confidence?
7. Are there alternatives?
Yes
No
8. Set NLG label associated to a message that prompts the user if he meant the alternative
9. Set NLG label associated to a message that teaches the user what the system is expecting
5. Change state to Low_Confidence and set the NLG label to prompt the user if he meant what was recognized
Yes
No
6. Generate the appropriate event in the FSM and set the appropriate NLG label, considering the scenario
10. Return NLG label

Figure 15: Dialog Manager's Main Method Steps Since the Reception of Recognized Text and Semantic

Secondly, the DM method checks the current state and each possible transition (cases 2. and 3.). These analyses were made in an exhaustive manner because the only available information (concerning the SCXML) that could be read by the C#, via the use of SCION, was the current state of the state chart. This meant that: all scenario specific calculations, the set of possible Phoenix slots for each state that can trigger state transitions, and the NLG labels to be returned needed to be defined in the C# side of the code, instead of, for instance, inside the SCXML file. Therefore, all verifications were made by hand on the C# side, inside the DM main method. An example of a simple state chart to support this explanation can be seen in Figure 16, with the corresponding verifications that were made in the code. The state "`Start`" is defined with two transitions. If an event of type `confirmation` is triggered, the FSM transits to "`State1`". If an event of type `denial` is triggered, the FSM transits to "`State2`".



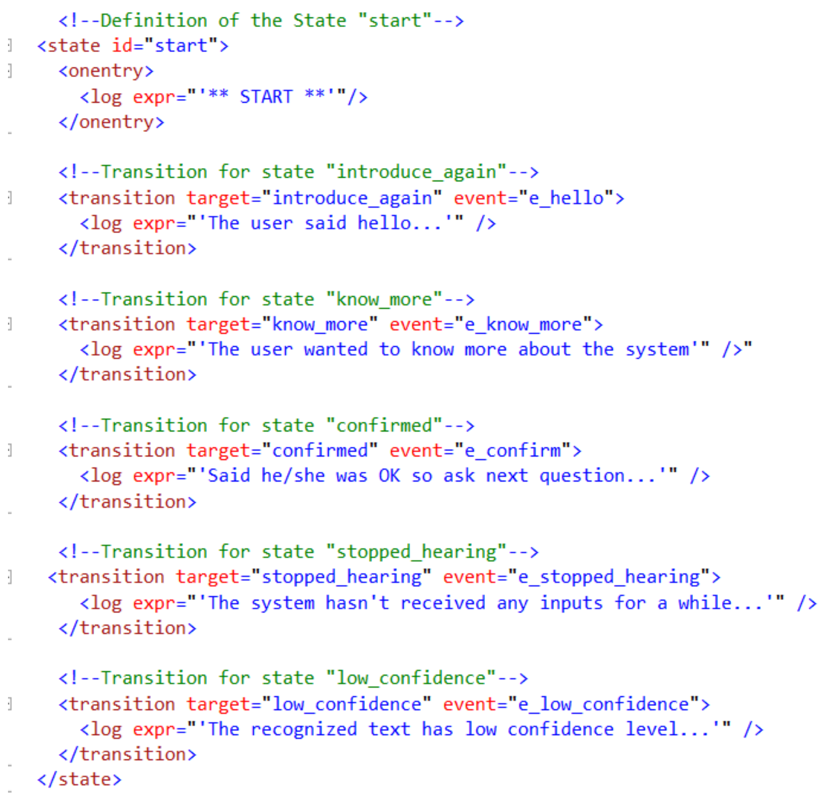
```
if(currentState.Equals("Start"){
    if(recognizedSemantic.Equals("[confirm]")
        generate("confirmation");
    else if(recognizedSemantic.Equals("[no]"))
        generate("denial");
} else if(currentState.Equals("State1")){
    (...)
}
```

Figure 16: Example of a Simple State Chart

The implemented DM solution needed to verify **by hand**, firstly, if the current state was "`Start`", "`State1`" or "`State2`". Secondly, considering the current state was "`Start`", the implemented solution needed to verify **by hand** if the received NLU slot was equal to one of the events that can trigger transitions either to "`State1`" or "`State2`". If the NLU slot was *[confirm]*, then the C# would generate (by hand) the event "`confirmation`" and the FSM would transit to state "`State1`".

Continuing the analysis of Figure 15, if the recognized semantic is indeed equal to one of the possible transitions for the current state then the confidence of the recognition is analyzed. If there was low confidence in the recognition, one of the special FSM states is triggered, the "`low_confidence`" state (this situation is represented with a thicker limit line, on case 5.) and the NLG label to be returned by the DM corresponds to a message that prompts the user if he really meant what was recognized. On the contrary, if the recognition was made with a strong confidence, then the appropriate event in the FSM is generated and also the appropriate reply for the scenario is returned (both defined by hand on the DM's C# side). What is called an appropriate event to be generated in the FSM depends on the current state. The appropriate reply depends on the transition. These values, clearly, depend on the scenario of the dialogue.

An example of a state definition in SCXML is presented in Figure 17. The figure illustrates the syntax for creating a more complete "`start`" state and the events that trigger new transitions to other states. For example, the event "`e_hello`" will trigger a transition to the state "`introduce_again`". Figure 18 illustrates the same information but in the graphic form, considering the FSM structure already explained. If the current state is "`start`" and the received semantic is a confirmation after a robot introductory message that asks if the user is ready for a game, then the event to be generated on the FSM is “`e_confirm`” and maybe an appropriate reply would be a message that says that the robot is glad to know it and that prompts the user for the next question.



```
  <!--Definition of the State "start"-->
<state id="start">
  <onentry>
    <log expr="'** START **'"/>
  </onentry>

  <!--Transition for state "introduce_again"-->
  <transition target="introduce_again" event="e_hello">
    <log expr="'The user said hello...'" />
  </transition>

  <!--Transition for state "know_more"-->
  <transition target="know_more" event="e_know_more">
    <log expr="'The user wanted to know more about the system'" />"
  </transition>

  <!--Transition for state "confirmed"-->
  <transition target="confirmed" event="e_confirm">
    <log expr="'Said he/she was OK so ask next question...'" />
  </transition>

  <!--Transition for state "stopped_hearing"-->
 <transition target="stopped_hearing" event="e_stopped_hearing">
    <log expr="'The system hasn't received any inputs for a while...'" />
  </transition>

  <!--Transition for state "low_confidence"-->
  <transition target="low_confidence" event="e_low_confidence">
    <log expr="'The recognized text has low confidence level...'" />
  </transition>
</state>
```

Figure 17 - Example of a State Definition in the SCXML Syntax

However, if the recognized semantic does not match any of the possible FSM transitions on the current state, the alternatives are checked. On one hand, **if no alternative exists**, then the DM returns a label that corresponds to a message in the NLG which means to **“teach” the user** what the system is expecting at that dialogue context. This can be done either by giving examples of what the user can actually say or via more indirect methods, such as giving hints, so the user shall know what set of words/sentences to say and what the system is expecting. **On the other hand**, the NLG message to be synthesized prompts the user **if he meant the alternative word, because there is a high probability that the alternative word makes sense for the current dialogue context**.

To conclude, the states and the events that trigger transitions were defined in the DM’s SCXML side, while the event sending mechanism, as well as the comparison between the NLU slots and the events that can trigger transitions inside each state was defined in the DM’s C# side.

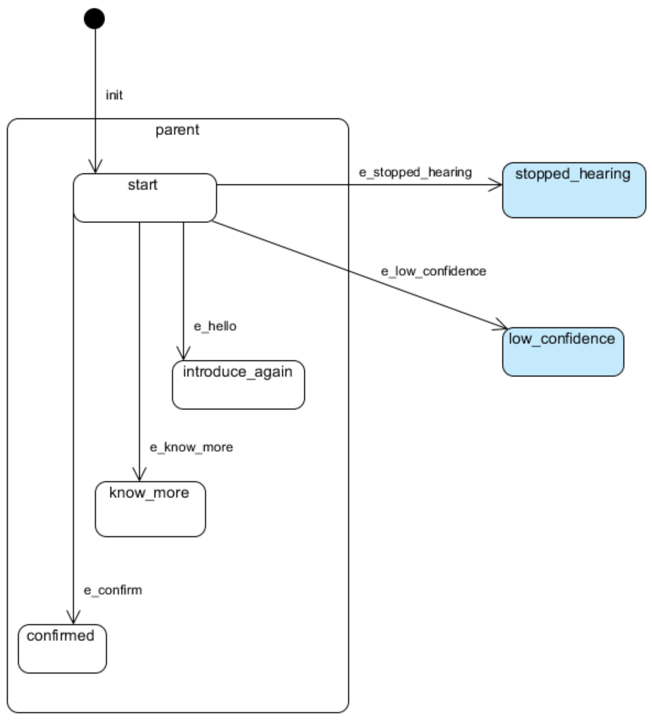


Figure 18: Example of a State Definition (Graphic form)

3.2.2.4. Natural Language Generation

This is the module responsible for the return of a text message to be synthesized. To contextualize, at this moment the DM has returned a label that identifies the type of message that the NLG needs to fetch. The syntax of this label can be seen further in this section. The Related Work analysis has led to the conclusion that the most typical type of NLG is based on templates or canned text sent directly from the DM. **The implemented NLG is template-based**. The DM, after analyzing the recognized text and semantic and the dialogue state, delivers to the main server program a NLG label. The main server program then sends this label to the NLG module. This label can have special tags attached which replace special words in the NLG template, when necessary. The attached words are named `TAGx` where `x` is a number, and they come connected to the NLG label with the following syntax:

```
[SLOT]:TAG1_TAG2_...TAGx
```

This format is useful, for instance, when the user gives some specific information to the system and the system wants to give feedback on what it understood. So, for instance, in the "`low_confidence`" special state, the DM returns a label with the following format:

```
"[low_confidence_level]:" + result.Text
```

Where "`result.Text`" is the recognized text sent to the DM module, obtained by the ASR. In this case, the NLG label `[low_confidence_level]` will correspond to a sentence with a member called `TAG1` that will be replaced by the recognized text.

The NLG uses a configuration file to retrieve the appropriate reply. The format of this configuration file can be seen in Figure 19, where each sentence was written twice: once in Portuguese and once in English.

```
low_confidence_level_pt
    Foi TAG1 que disseste?
    Disseste TAG1?

low_confidence_level_en
    Was it TAG1 that you said?
    Did you say TAG1?
;

say_goodbye_pt
    Gostei muito da tua companhia! Até à próxima!
;
say_goodbye_en
    I enjoyed a lot your company! See you next time!
;
```

Figure 19: Example of the NLG configuration file

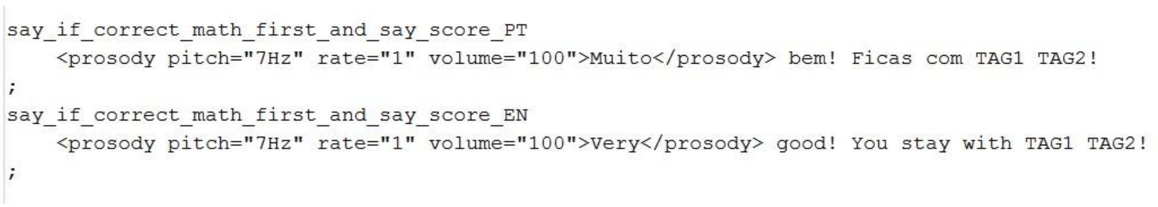
It can be seen the `[low_confidence_level]` example with the `TAG1` element. Each NLG label may also have one or more possible sentences. This happens in order to create a more comfortable interaction so the users do not easily get bored of the system. In addition, other replies are used to teach the user what to say in case nothing from the awaiting semantic was recognized. In Section 3.2.2.3 it was mentioned that, in this specific case, as no transition occurred and the FSM maintains its current state, then the adopted solution "teaches" the user what the system is expecting. This is done by using the same NLG label and by fetching a new sentence. There are also special characters that can be used before the name of the NLG label that allow for repetition (#) of a reply or the reset into the first reply (&). These are useful whenever the user asks for the system to repeat what was previously said, for instance. **If no special character is defined** by the

DM, then **the NLG incrementally returns the next sentence** of a given label, restarting when it reaches the last one.

Another important aspect of the NLG is the use of SSML (Speech Synthesis Markup Language) to give some emphasis to the reply when it is synthesized [55]. This language was used because sometimes the synthesizer did not pronounce the question and exclamation marks. In addition, by giving some emphasis to words such as "**`very`**" ("`muito`" in PT) and "**`nothing`**" ("`nada`" in PT) and so on, the robot would sound more empathetic with the user. This language supports many elements such as:

- The "`emphasis`" element that immediately emphasizes a given word or sentence;
- The "`break`" element which allows for the inclusion of a break in the middle of a sentence;
- The "`prosody`" which allows for a change in the utterances' pitch, rate and volume.

The "`break`" element is very useful when some sentences do not include important pauses when they are synthesized. Unfortunately, the most interesting element which is the "`emphasis`" element is not currently supported by the speech synthesis engines for Windows and setting values for its attributes will produce no audible change in the synthesized speech output [56]. Therefore, in order to give some emphasis in utterances that involved questions and some exclamations, the "`prosody`" element was used. Figure 20 illustrates an example with a NLG label that involves SSML in the replies, also with Portuguese and English cases.



```
say_if_correct_math_first_and_say_score_PT
    <prosody pitch="7Hz" rate="1" volume="100">Muito</prosody> bem! Ficas com TAG1 TAG2!
;
say_if_correct_math_first_and_say_score_EN
    <prosody pitch="7Hz" rate="1" volume="100">Very</prosody> good! You stay with TAG1 TAG2!
;
```

Figure 20 - Example of a slot in the NLG with SSML

An increase of the pitch was used whenever words such as "**`very`**" ("`muito`" in the example) and "**`nothing`**" ("`nada`") were used. A value of 7Hz was concluded to create the best results. The rate was rarely modified, being its use related to some rare cases where the words were being pronounced too fast or too slow than the normal human utterances, or in "repeat" cases, slowing down a little the returned sentence.

3.2.2.5. Speech Synthesis

This module is responsible for the synthesis of the NLG sentence. At this moment, the main server program knows the NLG message to be synthesized. It was already mentioned in previous chapters that both the recognizer and the synthesizer are from Microsoft^{TM}. The only supported gender and voice is the Microsoft Server Speech Text to Speech Voice (pt-PT, Helia) Portuguese voice. This module transforms the reply in an audio format with 1600 samples per second, sixteen bits per second and with mono channel. This audio is then sent to the client to be played by the robot’s default playback audio device.

3.2.2.6. Observations

Considering the Related Work analysis, the implemented SDS is above the general developed ones in two specific characteristics: **it considers some empathetic interaction by using SSML in the NLG messages and it is developed in the Portuguese language.**

Moreover, the SDS DM module does not stay behind the normal characteristics: it has low confidence handling by making sure that what the system recognized is exactly what the user said; it has interruption handling by the use of “stop” commands; and it has out of domain utterances strategy by analyzing alternatives and teaching the user what the system is currently waiting for. However, it is easy to conclude that this is a very extensive solution for the DM as every state and every transition was analyzed in the C# by hand, because the awaiting semantic and other relevant information were not defined anywhere in the code. The code resulted, then, in around **1200 lines for the DM’s C# file**. This solution **was not flexible** for other scenarios and it was almost impossible to integrate with other robotic components. Therefore, a new solution was implemented: based on the Information State dialogue management. This solution can be seen in Section 3.3.

3.3. A Second Version of the System

It is normal at the development of any system to mold it in order to better fulfill the requirements and objectives of such system, especially after the testing phase. The main improvements which were made focus: on completing the grammars for a more natural interaction, on testing better "`prosody`" attributes for the NLG sentences to create the ideal empathy towards users and on a new type of dialogue management. The dialogue management was initially based only on a FSM, with a structure that considers a "`parent`" state which includes the scenario specific states. Outside of the "`parent`" state there were the special states definitions for special interaction cases that do not depend on the scenario. Nevertheless, this **solution was not flexible enough for future scenarios and deployment in robotic platforms**, which resulted in the adaptation of another solution. The **new solution results** in the fact that, if a set of rules are respected, **no more modifications** are **needed by future developers**, **despite new words/phrases for the ASR grammar and new sentences and labels in the NLG component**. This was reached by creating a DM that uses both the **Information State type and the FSM**.

3.3.1. Dialogue Management – Second Solution

The previous solution was a very extensive solution, as every state and every transition was analyzed in the DM's C# side by hand (Section 3.2.2.3.). These analyses were made in an exhausting manner because the only available information (concerning the SCXML) that could be read by the C#, via the use of SCION.NET, was the current state's name on the state chart. This meant that: all scenario specific calculations, the set of possible NLU slots (that define the semantic of the recognized text) for each state that can trigger state transitions, and the NLG labels to be returned after each transition needed to be defined in the C# side of the code, instead of, for instance, inside the SCXML file, in order to reduce the C# overload. Therefore, all verifications were made by hand on the C# side, inside the DM main method, leaving only the states and transitions declaration for the SCXML side.

It is relevant to **facilitate future work for developers**, especially when one of the objectives of this dissertation is to **provide a flexible solution that supports future research and development of Spoken Dialog Systems for robots**. Considering this requirement, there was the urgent need to search for a more flexible solution. Firstly, Figure 21 illustrates the communication between the C# and the SCXML in the previous implementation. It can be seen which component performs what operation/verification.

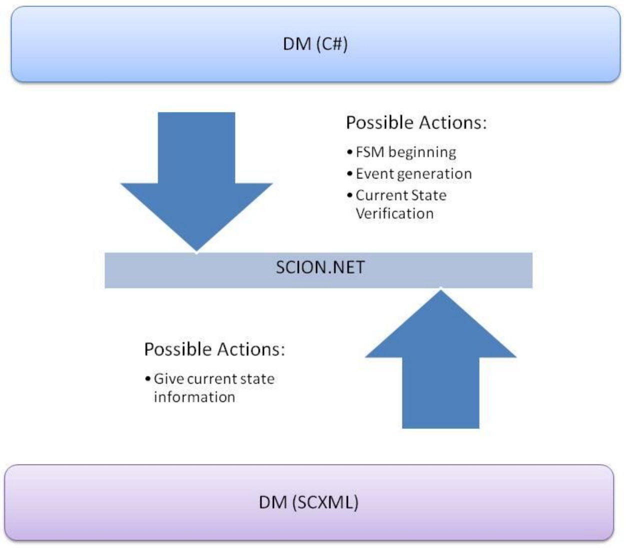


Figure 21: Previous C#-SCXML Communication Model

The new solution was possible by finding a way to declare on the SCXML side a set of calculations and declarations that would reduce the DM's C# overload side. For instance, the awaiting semantic for each state, the proper NLG label to return after each transition and the scenario specific calculations, if implemented on the SCXML side, leaving only the event generation and the special states handling for the C# side, would allow for a more flexible solution. Figure 22 illustrates this new approach. Also, by knowing the awaiting semantic for the current state on the FSM and the NLG label that each transition sets whenever an event is generated makes it easier for the DM's C# side to create a more general solution. Figure 23 illustrates the new verification process for the simple case already presented on Figure 16.

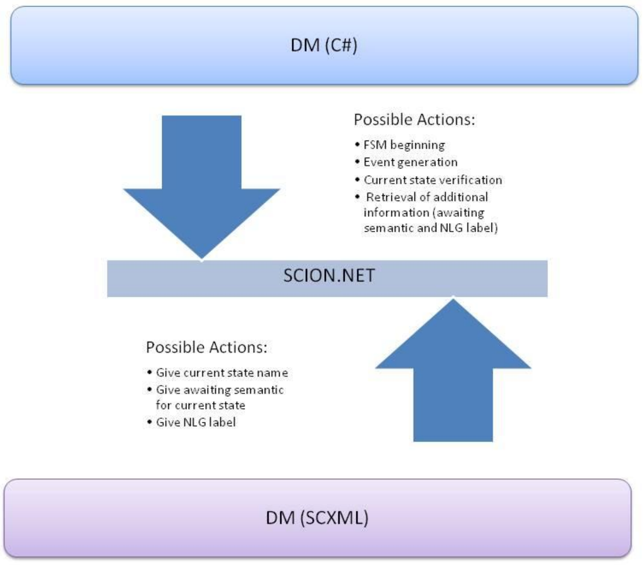
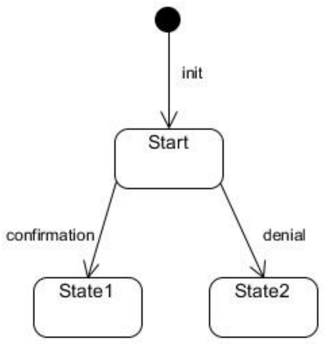

DM (C#)
Possible Actions:
• FSM beginning
• Event generation
• Current state verification
• Retrieval of additional information (awaiting semantic and NLG label)
SCION.NET
Possible Actions:
• Give current state name
• Give awaiting semantic for current state
• Give NLG label
DM (SCXML)

Figure 22: The new C#-SCXML Communication Model



```
/* Current state is Start.
 * recognizedSemantic is "confirmation" */

if(awaitingSemantic.Contains(recognizedSemantic))
    generate(recongizedSemantic);

/* Current state is State1. */
```

Figure 23: Example of Simple State Chart (new solution)

The awaiting semantic array and the NLG label are part of the **Information State** dialogue management type, and they are detailed in Table 8.

SCXML is a very complete language for state charts, so it was easy to find a solution that allows for these declarations, via the use of the "`script`" element that allowed for Javascript code to be included. The problem, however, was to create an interface that would permit the communication between the C# and the SCXML, apart from the event generation and current state name retrieval which was already possible with the available SCION functions. The solution consisted in the addition of a SCION method which registers functions to be called inside the SCXML, stating the methods' arguments and return values. These registered functions would be the desired interface. An inner public helper class was created to define these functions to be registered, as well as the **Information State** variables for the new DM solution. These functions are described in Table 7 and the Information State variables can be seen in Table 8. While observing Table 7, it is important to note that the arguments represent values stored/used on the DM's C# side and the return values represent values that are stored/used on the DM's SCXML side.

Function Name	Purpose
`void setMaxValuesForInteraction(int maxRepetitions, int maxInitiatives, int maxLowConfidence)`	To set the max values for interaction. `MaxRepetitions` is the max repetitions that the server can perform before a reset in the dialogue. `MaxInitiatives` is the max server initiatives in the dialogue before a reset. `MaxLowConfidence` is the max number of low confidence recognitions before a reset.
`void setReply(string newReply)`	To set the NLG reply after some transition has occurred. It also sets the last NLG reply that is used in repeat requests by the user or to teach the user what to say, since the NLG label can have multiple sentences.
`string getRecognizedSemantic()`	Used by the SCXML to get the recognized semantic.

`string getRecognizedText()`	Used by the SCXML to get the recognized text.
`void setAwaitingSemantic(string awaitingSemantic)`	To set the awaiting semantic for the current state.
`void setErrorProneWords(string errorProneWords)`	To set the error prone words. These words vary from language to language and are error prone or can be misunderstood in the recognition process. They are used to define whether there are alternatives for the recognition.

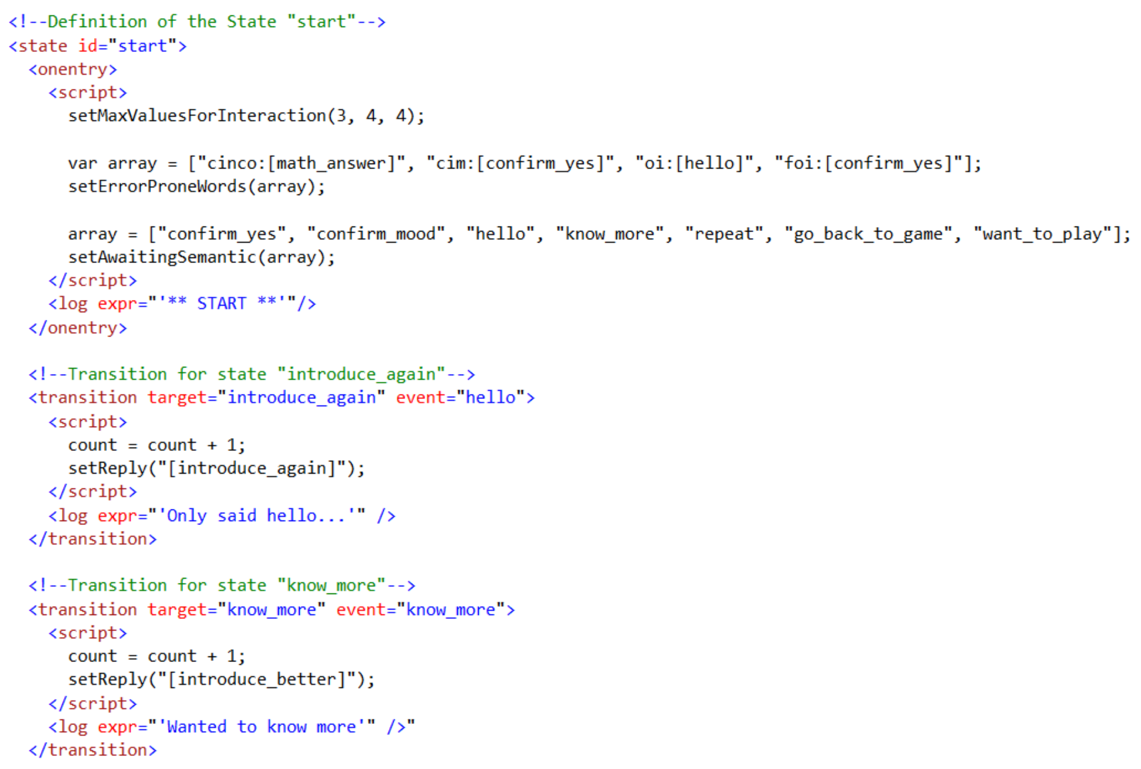
Table 7: Registered Functions for the SCXML-C# Interface

Information State Variable	Definition
`lastReply`	List with the last NLG label given by the system. It is a list because sometimes two replies are given at a time (depends on the scenario).
`recognizedSemantic`	The semantic of the recognized input. This is the Phoenix main slot.
`recognizedWords`	The text of the recognized input.
`confidenceLevel`	The confidence in the recognition process.
`currentStates`	The current state of the FSM.
`lastState`	The last state where the FSM was.
`errorProneWords`	A list of error prone words. Each language has its examples, such as "**`cinco`**" (`five`) and "**`sim`**" (`yes`) in Portuguese. Useful for the seeking of alternatives. Format: `["word1:[PHOENIX_SLOT_word1]", "word2:[PHOENIX_SLOT2_word2]",…, "wordn:[PHOENIX_SLOTn_wordn]"]`
`hasAlternative`	If `recognizedWords` contains alternatives/error prone words. Used when nothing expected was recognized.
`misunderstandedWord`	This holds the alternative word chosen to be prompted to the user.
`totalRepetitionsMade`	Total number of repetitions made by the system. Suffers a reset on each interaction (new dialogue session or whenever the "`start`" state is called).
`totalLowConfidenceInputs`	Total number of low confidence inputs. Suffers a reset on each interaction (new dialogue session or whenever the "`start`" state is called).
`totalSystemInitiatives`	Total number of system initiatives in the dialogue. Suffers a reset on each interaction (new dialogue session or whenever the "`start`" state is called).
`awaitingSemantic`	A list of the awaiting semantics for the current state. Changes in every transition for a new state. Format: `["PHOENIX_SLOT1", "PHOENIX_SLOT2", …, "PHOENIX_SLOTn"]`

`predictedEvent & predictedReplies`	Used when an event occurs in the FSM that makes it go out of its main body and needs to know the predicted event and replies that would had taken place if everything was normal. Used for `“stopped_hearing”`, `“low_confidence”` special states.

Table 8: Information State Variables

By defining these functions, it was possible to create a more flexible solution. First of all, **all the states declaration** (inside the main body of the scenario and the special states) **were maintained as well as their architecture inside SCXML, but each state's definition started to include Javascript code** that allows for an array of awaiting semantic to be set. Also, **on each transition's definition there is also Javascript code** which allows for the setting of the NLG label or labels to be returned by the DM. This new syntax can be seen in Figure 24, which illustrates the new definition of some of the states in Figure 17 and Figure 18. It can be observed the use of some functions described in Table 7.



```
<!--Definition of the State "start"-->
<state id="start">
  <onentry>
    <script>
      setMaxValuesForInteraction(3, 4, 4);

      var array = ["cinco:[math_answer]", "cim:[confirm_yes]", "oi:[hello]", "foi:[confirm_yes]"];
      setErrorProneWords(array);

      array = ["confirm_yes", "confirm_mood", "hello", "know_more", "repeat", "go_back_to_game", "want_to_play"];
      setAwaitingSemantic(array);
    </script>
    <log expr="'** START **'"/>
  </onentry>

  <!--Transition for state "introduce_again"-->
  <transition target="introduce_again" event="hello">
    <script>
      count = count + 1;
      setReply("[introduce_again]");
    </script>
    <log expr="'Only said hello...'" />
  </transition>

  <!--Transition for state "know_more"-->
  <transition target="know_more" event="know_more">
    <script>
      count = count + 1;
      setReply("[introduce_better]");
    </script>
    <log expr="'Wanted to know more'" />"
  </transition>
```

Figure 24 - New SCXML Syntax for the Definition of a State and its Transitions

It can also be seen the format for the definition of the error prone words and the awaiting semantic, as defined on Table 8. Other functions such as `getRecognizedSemantic` and `getRecognizedText` are only used when the FSM's current state needs to store some scenario value included in the recognized text (such as the user's age, for instance) in order to continue the dialogue. In addition, if the FSM transits to the "`low_confidence`" state, it must know the previous recognized text and use it if the user confirms what was said (this is part of the predicted event and reply IS variables presented in Table 8). The Information State variables are updated by both the DM's C# side and the DM's SCXML side, and their setting depends on Information State events which trigger Information State rules. **These updates were implemented with the consideration that future developers do not need to change these events and rules. However, if some changes are indeed needed, they are easy to be applied because of the developed C#-SCXML interface (the inner helper class already mentioned).** The update rules of the Information State variables can be seen on Table 9. But firstly, these are the Information State events that are used by the rules:

- `Timeout()`: when the system does not hear the user for a certain amount of time;
- `Repeat()`: when nothing was recognized from the ASR and the system asks the user to repeat;
- `Stop()`: when the user wants the system to stop playing the current reply.

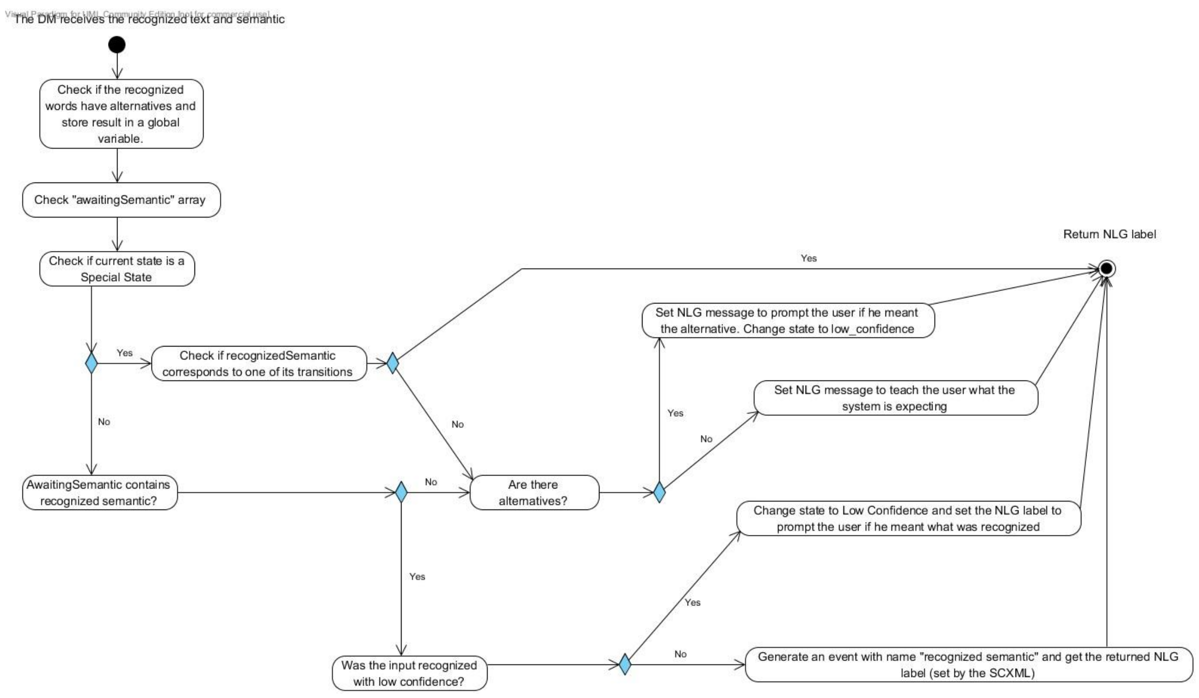
Rule	Definition	Conditions and Actions
Stop	**Stops speaking the current sentence**	**Conditions:** ✓ `Stop()` **Actions:** ✓ Sends a message to the client for it to stop playing the last returned audio stream.
AskIfOk	**Prompts the user if everything is OK.**	**Conditions**: ✓ `Timeout();` ✓ `totalSystemInitiatives < MaxSystemInitiatives`. **Actions**: ✓ Gives a NLG reply which prompts the user if everything is OK. Changes state to "`stopped_hearing`" and updates `awaitingSemantic`.
AskIfContinuesDialogue	**Prompts the user if he/she wishes to continue the dialogue**	**Conditions**: ✓ An interaction variable as reached its maximum value;

		✓ This situation has not occurred before. **Actions**: ✓ Changes state to "`bad_interaction`"; ✓ Gives NLG reply which prompts the user if he/she wishes to continue the dialogue; ✓ Resets the interaction variables.
GiveAnswer	**Gives a reply to the user based on the semantic received**	**Conditions**: ✓ Something was recognized from the ASR and NLU has returned its semantic; ✓ The recognized semantic is on the `awaitingSemantic` array; ✓ Confidence is ABOVE 70%. **Actions**: ✓ Generates an event with name of recognized semantic; ✓ Gives NLG reply set by the SCXML; ✓ Updates awaiting semantic.
GiveOtherAnswer	**Nothing expected was recognized. Teaches the user. No error prone words in the input.**	**Conditions**: ✓ Something was recognized from the ASR and NLU has returned its semantic; ✓ The recognized semantic is NOT on the `awaitingSemantic` array; ✓ The recognized words are not error prone; ✓ `totalSystemInitiatives < MaxSystemInitiatives`. **Actions**: ✓ Teaches the user what the system is expecting him to say.
GiveSameAnswer	**Gives previous answer when the user asks the system to repeat.**	**Conditions**: ✓ The user asked the system to repeat what was last said. **Actions**: ✓ Gives a NLG reply with the previous returned NLG label and the character # attached.
AskToRepeat	**Asks the user to repeat what he said**	**Conditions**: ✓ `Repeat();` ✓ `totalRepetitions < MaxRepetitions`.

		Actions: ✓ Gives NLG reply that prompts the user to repeat what he/she said.
ConfirmWhatUserSaid	**Asks the user if the system understood what was said.**	**Conditions**: ✓ Something was recognized from the ASR and NLU has returned its semantic; ✓ The recognized semantic is on the `awaitingSemantic` array; ✓ Confidence is BELOW 70%. **Actions**: ✓ Change state to "`low_confidence`"; ✓ Gives NLG reply that prompts the user to confirm if he/she said what was understood by the ASR; ✓ Updates the awaiting semantic.
CheckAlternatives	**Checks if the recognizedWords are error prone and checks alternatives.**	**Conditions**: ✓ Something was recognized from the ASR and NLU has returned its semantic; ✓ The recognized words are error prone. **Actions**: ✓ Checks if the error prone words have alternatives for the recognition that make sense for the current context; ✓ Sets global variable with the result.
AsksTheUserIfHeMeantSthElse	**Asks the user if he meant another word. Input contains error prone words.**	**Conditions**: ✓ Something was recognized from the ASR and NLU has returned its semantic; ✓ The recognized semantic is NOT on the `awaitingSemantic` array; ✓ The recognized words are error prone and there are alternatives for the recognized words. **Actions**: ✓ Gives NLG reply that prompts the user if he meant the alternative word.

Table 9 - Information State Rules

As a result, by having a more complete SCXML-C# communication, there were a few modifications in the DM's sequential processing. A new activity diagram for the DM's C# side can be seen on Figure 25. Generally, the DM first checks if the FSM is on a special state and deals with it accordingly, as before**. The processing of the special states is maintained in the DM's C# side because some states such as the "`stopped_hearing`" state are not triggered by user utterances and rather by system events, which makes their implementation easier and less error prone if these states are maintained in the C#.** Also, some states need more than one event to be generated, such as the "`low_confidence`" state, because this state makes use of the `predictedEvent` variable as described on Table 8. Apart from this, as they are generic to any scenario, this is also not very relevant for future developers, since **no modifications apart from grammar based are expected to be needed**. Either way, even if some modifications are needed for future developers, **these states' definition was facilitated by the use of an additional C# file where developers can define the main information in an easier way**.



Visual Paradigm for UML Community Edition [not for commercial use]
The DM receives the recognized text and semantic
Check if the recognized words have alternatives and store result in a global variable.
Check "awaitingSemantic" array
Check if current state is a Special State
Yes
No
Check if recognizedSemantic corresponds to one of its transitions
Yes
No
AwaitingSemantic contains recognized semantic?
No
Yes
Are there alternatives?
Yes
No
Set NLG message to prompt the user if he meant the alternative. Change state to low_confidence
Set NLG message to teach the user what the system is expecting
Return NLG label
Change state to Low Confidence and set the NLG label to prompt the user if he meant what was recognized
Was the input recognized with low confidence?
Yes
No
Generate an event with name "recognized semantic" and get the returned NLG label (set by the SCXML)

Figure 25: Dialog Manager's Main Method Steps Since the Reception of Recognized Text and Semantic (new solution)

Secondly, if the FSM is not on one of the special states, the DM reads the awaiting semantic for the current state and checks whether the Phoenix main slot retrieved from NLU is included in this array. Finally, if the recognized semantic is included on the array and the recognition confidence was good then the **DM generates an event with the same name as the**

recognized semantic, to turn the solution general. This can be seen by observing Figure 24 where the following names represent Phoenix main slots: "`confirm_yes`", "`confirm_mood`", "`hello`", and so on. After generating an event, the DM reads the new NLG label which was set on the SCXML by the use of the "`setReply`" method, to send it to the main server program. This makes the solution very easy to use and adapt for other scenarios. Also, the new solution resulted in about **380 lines** of code on the DM's C# side, which is an enormous difference comparing to the **previous 1200** lines.

The "`bad_interaction`" Special State

The observation of Table 7 and Table 8 concludes that there are additional interaction values that will trigger special messages or even a reset on the system when they reach their defined maximum value. When **one of these variables reaches its maximum value**, then the **FSM goes to one additional special state**: "`bad_interaction`", and **that variable suffers a reset**. The transitions to this state are handled on the DM's C# side as not all interaction variables are incremented by user utterances (same logic as why the special states handling was maintained on the C# side). When it is the first time that the FSM transits to this special state, the system first waits for a confirmation or a denial from the user indicating if he/she wishes (or not) to continue the interaction. If the user wants to continue the interaction, the FSM transits back to the state it was (using the "`history`" element, as explained before). Consequently, if the FSM transits once more to this special state, the dialogue suffers a reset and the DM sends a special NLG label with a sentence indicating that the robot is sorry for the inconvenience, but the conversation has run too confusing and it is better to start again. **All interaction variables suffer a reset whenever the "`start`" state is called**. It would be better if other types of reset were possible, but it was not implemented.

Rules for Future Developers

Future developers are invited to respect the following rules:

- The main body of the FSM should be maintained so the special states are defined outside of the "`parent`" state, being the main body dedicated to scenario specific states and transitions;
- All transitions to the special states and the rules that set when they are triggered should be maintained both on the SCXML and C#;

- When creating a new state, there must be a "`script`" element where the `setAwaitingSemantic` function is called to define that state's awaiting semantic;
- When adding transitions, there must be a "`script`" element where the `setReply` function is called to set the NLG label;
- At the first state of the "`parent`" state, there should not only be a "`script`" element indicating the awaiting semantic but also the error prone words and the max interaction values as shown in Figure 24.

3.4. Final Remarks

The first implementation of the system was created in order to have a complete and functional system. The stop mechanism on the user perspective was included, as well as a complete adaptation to the Portuguese language with a better synthesized voice. Later then, the system was molded in order to facilitate future work for the developers. A new DM was implemented that satisfies the main objectives for this dissertation. Future developers have an implemented interface that allows for them to use functions inside the SCXML without worrying about the DM's C# side calculations. They can even easily add and register new functions if necessary. The only necessary effort stands in the development of a FSM, according to the scenario they choose, and the definition of the ASR, NLU and NLG grammars.

Considering both solutions and the Related Work analysis, it can be concluded that the DM does not stay behind the average DM characteristics in the following aspects: it has low confidence handling by making sure that what the system recognized is exactly what the user said; it has interruption handling by the use of "stop" commands; it has out of domain utterances strategy by analyzing alternatives and teaching the user what the system is currently waiting for; it has similar methodologies for the NLU and NLG components; it allows somehow for mixed initiative dialogues **and, because of the new solution, it is flexible for other domains**.

The SDS is above the general developed ones in the following characteristics: it considers some empathetic interaction by using SSML in the NLG messages, it is developed in the Portuguese language and **the system takes actions according to some interaction values so it does not become tiresome**.

The SDS is below the average in the following specific characteristics: it does not consider user interpolation, since no data is stored permanently; and it is not domain switching. However, it can easily become domain switching. This will be described in Chapter 5.

To conclude, the implemented DM does not stand behind the Related Work analysis. The next step to take was to evaluate the system. It is relevant to evaluate the usability of the system on both user interaction and developers perspective; however, it is very difficult to evaluate the overall usability of the developed system because the main objective was to have a base implementation that gives continuity to the research and development on Spoken Dialog Systems for the Portuguese voice. Also, the system is scenario independent. A test scenario was chosen to get some usability evaluation results, and the evaluation on the developer's perspective was also made. This is detailed on the next chapter, Chapter 4.

Chapter 4: Evaluation and Results

The architecture and the implementation of the developed Spoken Dialog System have both been described in Chapter 3. In order to test it, a scenario was chosen considering the domain of service and interactive robots, which will be explained in the following section. Some people were asked to test the usability of the system on the user interaction and developer's perspectives and the evaluation proceedings and results are described on Section 4.2.

4.1. Test Scenario

The Robocup@Home is the largest international annual competition that encourages teams from all over the world to participate in many challenges that may prove their innovative contributions in the area of autonomous service robots [35]. The main scenarios are home environment and, therefore, one of the main research fields is HRI and Cooperation. In the 2013 Robocup@Home Rules & Regulations [35], in the Stage II Demo Challenge, one of the focuses was "playing with children or baby-sitting". This topic is very appealing and also brings many challenges towards HRI. The implementation of a dialogue system within a robot in order for it to communicate with children in a human-human way may be very relevant for the support of children with impairment, such as children with autism. For instance, the AuRoRA project (AUtonomous RObotic platform as a Remedial tool for children with Autism) aims for the use of robots in order to help children with autism develop their social skills [57]. However, children can easily feel worn out, especially those who need special attention and care. But on the other hand the ideal approach gives children with impairment, who may have difficulty in expressing feelings and thoughts in words, chances to express themselves and therefore develop the social skills they so often lack [58].

As a result, this topic encourages the researchers to create the ideal empathy so that the children would not be easily worn out of interacting with the system, and opens a wide range of interesting scenarios that may change how children communicate every day. That's why the **chosen test scenario** for the implementation was **playing a mathematics game with children**. The game consists of five questions which grow more difficult. The initial scenario was also thought to include a general knowledge questionnaire for topics such as animals, objects and so on, but it was not implemented since the most relevant aspect was to test the overall interaction and not exactly to create a robust system in terms of this concrete scenario.

The scenario involves children from five to twelve years old. According to the age, the system chooses from sums or multiplications questions. From five to eight years old, the system asks sums questions, while from nine to twelve years old, the system asks multiplications questions. The scenario can be better observed through the state diagram in Figure 7. It is important to note that this diagram does not consider the special states (see Section 3.2.2.3.). First of all, the transitions on the state diagram represent the semantic of the recognized user utterances, given by the NLU, and the entry operations inside each state represent the label given to the NLG by the DM in order for the system to send an appropriate NLG reply. By observing the diagram and according to what was previously said about the client-server architecture, it is known that first the robot gives an introductory message to the user and then waits for some input. This input can be either a confirmation that everything is indeed OK; it can be another introduction such as "hello!" (olá, in PT), or it can even be a "know more" question where the user wants to know more information regarding the robot and the purpose of the conversation. Either one of these inputs can lead to another state where the user is prompted about his/her age. When the age is understood by the robot, then it asks the user if he/she is prepared for a mathematics game. If the answer is affirmative, then the rules of the game are explained, being followed by the game itself if there are no doubts. The game phase is the most complex of the scenario as it involves additional calculations and a variety of NLG labels, depending on: the current question; current score; and if the answers are correct or wrong. The diagram illustrates that the child can cancel the game at any time, but if that does not happen, then the "`end_game`" state will be reached when the answer for the final question (question number five) is given. Regardless of the situation that triggered the "`end_game`" state, the robot informs the child of the final score and asks if he/she wishes to play again, transiting to the "`say_rules`" state in an affirmative case. Otherwise, the robot says goodbye, the state machine goes back to its initial state and the robot waits for other user inputs.

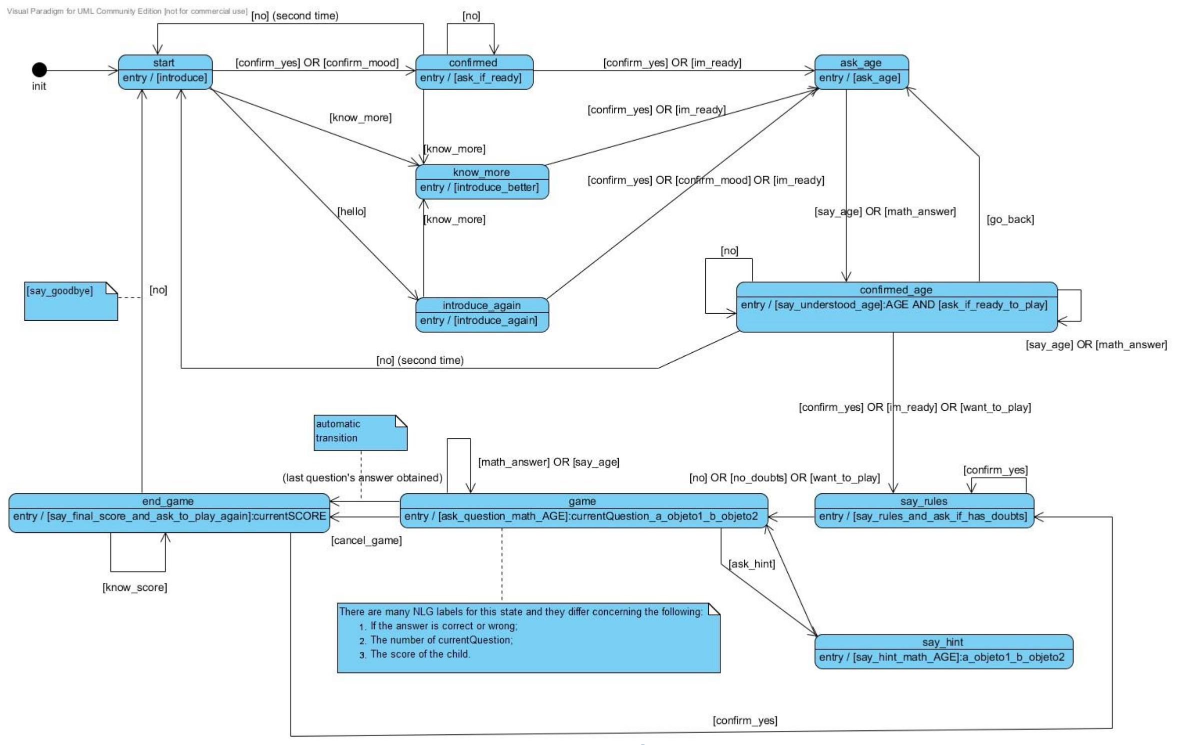

Visual Paradigm for UML Community Edition [not for commercial use]
[no] (second time)
[no]
init
start
entry / [introduce]
[confirm_yes] OR [confirm_mood]
confirmed
entry / [ask_if_ready]
[confirm_yes] OR [im_ready]
ask_age
entry / [ask_age]
[know_more]
[confirm_yes] OR [im_ready]
[know_more]
know_more
entry / [introduce_better]
[confirm_yes] OR [confirm_mood] OR [im_ready]
[hello]
[know_more]
[say_age] OR [math_answer]
[go_back]
[no]
[say_goodbye]
[no]
introduce_again
entry / [introduce_again]
confirmed_age
entry / [say_understood_age]:AGE AND [ask_if_ready_to_play]
[say_age] OR [math_answer]
[no] (second time)
[confirm_yes] OR [im_ready] OR [want_to_play]
automatic transition
[math_answer] OR [say_age]
(last question's answer obtained)
[no] OR [no_doubts] OR [want_to_play]
[confirm_yes]
end_game
entry / [say_final_score_and_ask_to_play_again]:currentSCORE
game
entry / [ask_question_math_AGE]:currentQuestion_a_objeto1_b_objeto2
say_rules
entry / [say_rules_and_ask_if_has_doubts]
[cancel_game]
[ask_hint]
[know_score]
There are many NLG labels for this state and they differ concerning the following:
1. If the answer is correct or wrong;
2. The number of currentQuestion;
3. The score of the child.
say_hint
entry / [say_hint_math_AGE]:a_objeto1_b_objeto2
[confirm_yes]

Figure 26- State Diagram of the Test Scenario

It can be concluded that the scenario can be divided in seven main phases:

1. Introduction
2. Ask for Age
3. Prepare Game
4. Game
5. End Game
6. Repeat game
7. Farewell

The mathematics questions are of the following format:

- Sum: Pergunta número TAG1. Se te der TAG2 TAG3 e alguém te der mais TAG4, com TAG5 ficas? (Question number TAG1. If I give you TAG2 TAG3 and someone else gives you TAG4, how many TAG5 do you keep?)
- Multiplication: Pergunta número TAG1. Se te der TAG2 TAG3 e multiplicares por TAG4, com TAG5 ficas? (Question number TAG1. If I give you TAG2 TAG3 and you multiply it by TAG4, with how many TAG5 do you keep?)

Being "TAGx", where x is a number, previously defined variables and randomly chosen for each question that are attached to the NLG label.

In order to have results on whether the system fulfills the objectives defined in Objectives, the usability of the system was evaluated in both interaction and developer perspectives. On one hand, on the usability evaluation on the interaction perspective, the verification of whether the HRI was intuitive and ideal was performed. Therefore, the following example questions are relevant:

1. Is the ASR grammar sufficiently robust for the scenario?
2. Is the SDS quick when processing inputs and giving an appropriate answer?
3. Are the NLG sentences adequate? Do they give the ideal empathy? Does this have any effect on the user side?
4. Does the user get easily bored when interacting with the system?
5. Does the user feel comfortable when interacting with the system?

On the other hand, on the developer perspective, the verification of whether the developed SDS was flexible for the implementation of other scenarios was performed.

4.2. Experiments

4.2.1. User Interaction Perspective

Firstly, the interaction between the robot and humans was evaluated. It is important to mention that only a specific test case was used for the usability tests, the test scenario described above. This happens because it was practically impossible to test the overall system, because of the SDS independence from the robot and the scenario. Considering the test scenario, the most interesting volunteers would be children. Two children were selected to perform the usability tests. Both tested the system at Universidade de Aveiro. The tests were performed in Portuguese and at an office environment and, because the system was not integrated in a robotic platform yet, both the client and the server were locally executed at a laptop over the office table. Some pens and paper were also put on the table to aid the children at the game stage.

The first child, of seven years old, tested the system on 29th May and the second, of seven years old, on the 3rd of June of 2014. Both received the following instructions (with respective translations):

Portuguese	English
Olá! Vais estar sentado à frente deste portátil. O computador vai começar a falar contigo e tu saberás quando deves responder. Sabes o que é um Kinect? (explicar, se necessário, e apontar para o Kinect em cima da mesa) Vais falar para este Kinect aqui.	Hello! You will be sited in front of this laptop. The computer will eventually talk to you and you will know when to answer. Do you know what a Kinect is? (explain if necessary and point the one on the table) You will talk to this Kinect here.
Vais jogar um jogo com o computador. O que achas?	You will play a game with the computer, what do you think?
Por favor, diz as palavras de forma clara e simples.	Please speak the words in a clear and simple way.
Tenta falar nem muito longe nem muito perto do Kinect.	Try to talk not too far nor too close from this Kinect.
Podes cancelar o diálogo quando quiseres bastando dizeres algo como “não quero falar mais” e podes pedir para repetir caso não tenhas percebido bem o que foi dito, com frases como “podes repetir?”.	You can cancel the dialogue whenever you want. You just have to say “I don’t want to talk anymore”. In other case, you can also ask for the computer to repeat what it previously said, by asking “can you repeat?”.
No fim, se tudo correr bem, gostaria que me ajudasses e me respondesses a algumas perguntas se não te importares! Podemos começar?	At the end, if everything goes right, I would like you to help me by answering a few questions if you don’t mind! Can you begin?

The system was then initiated if the child had no doubts and introduces himself as Hélia, beginning the dialogue. The dialogue phases were already explained on Section 4.1.

At the beginning of the experiment the child was intimidated and shy but she showed to be ready for a challenge. Unfortunately, the first experiment did not reach phase 3, Prepare Game, because the system was not able to recognize the child’s confirmations. The child repeated for some times the words “sim” (yes) but the system rarely understood and the child became bored. A few modifications were made in the system in order for it to begin at the game stage, but the problem persisted. The recognizer is not prepared for child utterances, and it could not understand any “yes” responses from this child. The numbers were not also correctly understood. Because the problem was on the recognizer, and not on the system, the issue was not so relevant because the focus was not on a robust recognizer. It was also advised for the child to speak in a clearer way but no improvements were made on the recognition and she got easily bored of the interaction and the experiment ended. No additional data was retrieved from the experience.

Concerning the first evaluation, a new approach was thought about. First of all, it was better to advise in future evaluations that the children can say other words in spite of “sim” (yes), such as “pode ser” (can be), “claro” (of course) and “OK”. Secondly, it was made an alteration on the system which involves the confidence level of the word “yes”. It started to always be considered as “high” so the children were spared of even more confirmations. The last adaptations are related to the ASR grammar and the NLG replies. Since children pronounce the word “yes” in a slightly different way as adults (instead of saying “sim” they sometimes pronounce it “sinhe”), then the ASR grammar started to include this option. Each NLG label related to the states where confirmations were needed started to include a sentence that, whenever the system understood something different from the context (which happened only a few times, when the child said a confirmation but the ASR interpreted as something else), it included words and sentences that the child could say at that stage. This was made in order to complete the out-of-domain utterances strategy, illustrated in Figure 15: Dialog Manager's Main Method , when no alternative is available. After endowing the system with these alterations, the second evaluation took place in the same conditions as the previous one.

The second evaluation went very well. The second child was not as intimidated as the first one and appeared very confident. The experience went through all the dialogue phases, however not sequentially, because the child had to repeat the experiment four times in order to pass through all game phases. Two of the repetitions involved the reset of the system for reaching one of the interaction limit numbers (the maximum number of low confidence inputs from the user). The first time this happens, the system prompts the user if he wishes to keep talking. The child always answered “yes” in these questions. However, at the second time this happens in the interaction, the system does not prompt the user and advises that it is better to reset the conversation. The first restart occurred before the game phase and the second within the game phase (phase 4). After these restarts, the child tried to go back to the stage he was because he showed a strong interest to start and finish the game with high score, but the system unfortunately did not support this option and the conversation became confused for the child, leading to other experiments. Other repeats involved the ASR’s weak capacity of recognition of child utterances, which led to a decision in giving small breaks from time to time to the child. This was very helpful in keeping the child motivated. In addition, at the middle of the evaluation the system was altered to begin at the start of the game to avoid the child to become too bored, since he had already made it through the first three phases.

It can be concluded that the recognizer indeed had strong difficulties recognizing the “yes” utterances as well, but as long as the child was told that he could confirm with other words such as “ok” and “of course”, the interaction improved reasonably. At the game phase, there were three times that the system understood a different answer than the answer given by the child, resulting in some frustration, but the child was encouraged by saying that he was right and that he was to be congratulated for knowing all the answers. This second experience went very well because of the positive attitude of the child. He seemed really comfortable throughout the experience and did not get too bored when the system did not recognize his sentences. At the end, the child accepted to answer a questionnaire consisting of ten questions, based on the System Usability Scale [59] and even gave some suggestions at the end of the evaluation. The results can be seen in Table 10. The possible answers were: totally disagree, disagree, don’t know, agree and totally agree.

Questions	Translation	Answer
Achas que gostarias de jogar este jogo mais vezes?	Do you think you’d like to play this game more times?	Totally agree.
Achas que a Hélia podia ser mais simples na forma como falou contigo?	Do you think that Hélia should had been more simple in the way she spoke with you?	Disagree.
Achaste fácil falar com a Hélia?	Did you think that it was easy to speak with Hélia?	Totally agree.
Achas que para jogar precisas da ajuda de um adulto?	Do you think you need help from an adult to play this game?	Totally disagree.
Achaste que o jogo foi bem feito?	Do you think the game was well made?	Agree.
Achaste que a Hélia foi confusa a falar contigo?	Did you think that Hélia was confusing while speaking to you?	Totally disagree.
Pensas que outra criança iria achar fácil falar com a Hélia?	Do you think that another child would think it is easy to speak with Hélia?	Totally agree.
Achaste cansativo falar com a Hélia?	Did you find tiring to speak with Hélia?	Totally disagree.
Sentiste-te confiante para falares com a Hélia?	Did you feel confident while speaking with Hélia?	Totally agree.

Precisaste de aprender a falar com a Hélia antes de conseguires jogar bem?	Did you need to learn how to speak with Hélia before being able to play correctly?	Disagree.

Table 10 - Questionnaire and Answers for the Second Experience

The answers show very good results. Also, they generate a System Usability Scale result of 92.5, which is considered above average [60]. This is also because it was a child answering the questionnaire, but at the end more questions were made in order to obtain more feedback from the experiments. It was asked if he liked to know Hélia and what he liked the most. He answered that he did like the fact that she introduced herself and also the game very well. After asking if he would change something in the interaction, he suggested that he would make more questions, especially Portuguese Language questions. It was also asked if he thinks the system could be used at school in order for other children to learn from the system, and he answered that it would be fun to use this system at school.

4.2.2. Developer Perspective

The main objective for this Dissertation was not to test the overall system, but to have a base version of it in order to conclude if the system was ready and apt for future reuse. After the evaluation of the interaction, other experiments began. These experiments' main objective was to collect the opinion of the participants on whether the implemented solution can be easily reused. One student and one former student from University of Aveiro volunteered for this task and they were both given the two developed implementations: the first one, when the DM's solution was not flexible; and the most recent one, with a new dialogue management type and the C#-SCXML interface. One of the main objectives was to verify if there were still redundancies in the most recent implementation of the DM, considering that the previous one had too many redundancies which avoided it from being a good solution. Another objective was to conclude if the most recent solution is indeed easy to reuse and adapt to any scenario.

Each volunteer installed the required technologies on their personal computers to successfully run the client and the server. The main developer rules were explained to each volunteer personally and via the use of a Power Point presentation, which the participants kept with them to consult when needed. Each volunteer chose two different scenarios. One student chose a simple scenario for University students that ask the robot for information regarding the different research institutes and departments. This volunteer performed the experience in six hours plus the time for the installation of the required technologies (which took two hours). After obtaining the results, the student concluded that the most recent solution actually has a fair decrease of redundancy in the DM, which facilitates the creation of scenarios without changing the implementation outside of the SCXML file. However, he said that there was still redundancy that could be solved, especially on the ASR and NLU grammar files. There is duplicated information that could be simplified by the generation of one grammar file into another, for instance. He also suggested that it would be interesting to have the most common NLU labels stored in a database with a complete set of sentences/words.

The student was asked if he had to change something in the DM's C# side, for instance, to add new functions to the developed interface. The answer was negative, which is a good result. In terms of easiness of use, the volunteer said that the creation of a new scenario is relatively easy to perform and, for simple scenarios, even a non experienced developer could have a working scenario within four to eight hours of work. He suggested, however, the previous definition of small pieces/blocks of code that can work as Lego pieces to develop/design a whole scenario. He thinks this would facilitate even more the work for future developers.

The second volunteer chose a restaurant scenario, where the robot knows the menu and can perform scheduling operations. This volunteer performed the experience in only four hours, including the installation time (around one hour), because unfortunately this experience could not be completed, due to lack of free time from the volunteer. The majority of the questions were asked and answered. He concluded that the new solution is very welcomed considering the previous one. He even commented that the fact that 1000 lines of code were taken from the previous solution, leaving the DM's C# side with 300 lines, makes a simple integration with the code on the SCXML side. He also commented that it is almost impossible to develop the chosen scenario on the previous DM implementation. He concluded as well that a conversion from the ASR grammar file into a Phoenix grammar file, or vice-versa, would be relevant future work to reduce duplicates and facilitate the reuse of the system.

The former student was also asked if he had to change something in the DM's C# side, and the answer was no, apart from a new path base for the server to read the configuration files. In terms of easiness of use, unfortunately this volunteer could not finish the scenario because of lack of free time. However, he believes that with the received explanations and with more time he would have certainly created an effective solution with the provided project.

Chapter 5: Conclusion

5.1. Work Overview

The main objectives of this Dissertation focused on providing an easy and flexible Spoken Dialog System in the Portuguese language that would support and give continuity to future research and development of spoken interfaces for robots. The system should be as much as possible robot independent to be deployed on any robot running any Operating System, and should allow for the adaptation to any conversation domain.

This solution started with the analysis of a previous implementation of a SDS with a client-server architecture that showed to be very promising since the concept allowed for its easy deployment on any robot. The architecture, in order to permit this robot independence, focused on the idea that the client runs inside the robot as an additional application while the server is remotely localized. A simple test scenario was made in order to test the implementation, where the dialogue management involved a simple Finite State Machine, created with the SCXML state chart language. After detecting the main improvements to be made, and taking advantage on a project for a Mobile and Intelligent Robotics course for a first experience, the solution started to include a stop command via voice (from the user perspective) which every human-robot interface must incorporate. A better Portuguese voice was also adapted into the SS module, from Microsoft^{TM}, and the SS started to consider SSML in the sentences to add some empathy to humans. The solution was also improved in order to cancel all the non necessary speech events (garbage speech events) for the dialogue context because they could bring inconsistencies to the dialogue management. In order to test this new implementation, a new scenario was chosen that involved an increase in complexity on the dialogue management (a more complex state chart) and the grammars became even more robust. During this course, the SDS was tested on a real robot, and the project continued to have positive results, especially much better results in the recognition (because of the garbage speech events removal) and synthesizing of sentences (because of the new Portuguese voice).

Later, at the end of the project, the solution was improved for a whole semester, as continuity to the Dissertation work. In order to decide the next project improvements, it was made an analysis of the literature and the state of the art on spoken interfaces for service robots. With this analysis it was possible to create a vision of what should be the next main objectives for each SDS module. Based on that, the main points to develop were selected. Information was gathered concerning the SDS components, their implementation and the most relevant characteristics of the developed systems, including evaluation results. There was a serious lack of information on the developed projects. Consequently, it was very hard to collect complete information. Nevertheless, relevant conclusions were taken for the next steps. The most important aspects to consider were the flexibility of the SDS and their domain switching capabilities. The previous project was not yet flexible for other scenarios, much less domain switching. Considering that most systems have interesting out of domain user strategies, due to their domain switching mechanisms, it was included a mechanism that informs the user what to say when nothing from the current context of the conversation was recognized.

A new dialogue management was also implemented which facilitates future work and makes it possible for an easy adaptation to any scenario/domain. The new dialogue management is based on a combination of a FSM and the Information State type, where the Information State variables and functions are defined within the SCXML. This reduced the dialogue management overload, greatly simplifying the definition of scenarios and the communication between the dialogue management sub modules. Therefore, an interface for communication between the SCXML and other dialogue management components was created, that can be easily modified by future developers if additional operations are needed, concerning the scenarios. This new solution contributed for the fact that future developers only have to define the state machines and add new words/sentences into the grammars, without being concerned with other components' functionality.

Other issues needed to be resolved. The speech recognizer easily confused a few words of the Portuguese language, so a strategy was developed considering alternative words and the confidence of the recognition. The synthesizing of the sentences was not working well for punctuation cases, so the use of the SSML was also improved. A new sequential dialogue management processing was also developed, considering strategies for no recognition cases (asking the user to repeat in a more intuitive way).

The developed SDS, after the adaptations, was not behind the studied research and development. It had the advantages of being developed in the Portuguese language and to include SSML on the synthesized sentences, to create more empathy towards the users. Also, it had the great advantage of being robot independent. The developed solution equals the majority of the analyzed characteristics, since it is flexible for other scenarios, allows for mixed initiative dialogues, adopts strategies to improve the interaction and uses a mix of FSM/IS dialogue management types. The only two characteristics from which the developed solution got behind were the permanent storage of user data and the domain switching capabilities. The permanent storage of user data makes more sense when deploying the system into a complete robotic platform, in order to relate the information with other robotic modules. For the dialogue to be domain switching, an approach has been considered and is explained in the future work comments.

After the adaptation and improvements of the project, the evaluation phase begun. The usability of the system was evaluated, both on the interaction and developer perspectives. A test scenario was chosen to evaluate the system on the interaction perspective and the main topic was playing a mathematics game with children. The experiences were made with two children, of seven years old, on different dates. Since the SDS was not deployed on a robotic platform and it made no difference if a robot were present in the experiments, the tests were made in an office environment, where the SDS ran locally in a laptop connected with a Kinect.

On the developer perspective, also two experiences took place. One students and one former student from the University of Aveiro volunteered to create different scenarios both on the first implementation of the SDS and on the most recent and more flexible one. The objective was to collect conclusions on the redundancy differences between the two solutions and if the most recent solution is indeed easy to reuse and adapt to a different scenario. One student chose a restaurant scenario and another chose a simple scenario where students ask the robot for information regarding the University's departments.

5.2. Main Results

This Dissertation resulted in the implementation of a base SDS that allows for the creation of dialogues in the Portuguese language, on any scenario. It also resulted in a dialogue management with the capacity to perform mixed initiative dialogues, with interruption and low confidence handling mechanisms, and out of domain strategies to improve the interaction.

The third result is the successful inclusion of some empathic/emotional interaction by using SSML for the sentences to be synthesized. The simulation of punctuation effects was successful and the sentences/words sounded more natural.

At least one child, although the difficulties on the recognition of child utterances, showed a very positive attitude while interacting with the system and could reach the final phase of the test scenario. The child had very positive answers for the questionnaire and even suggested that it would be interesting to use the system at school. This is a very important result considering the use of empathy in the synthesized sentences, and also a great achievement, considering the difficulties of child-robot interaction.

To conclude, one of the volunteers for the developer perspective usability evaluation said that the system improved reasonably from version one to the final version, and that any developer could create a simple size scenario within six-eight hours without much effort.

5.1. Future Work

Taking in consideration the results obtained during the evaluation phase, and the conclusions obtained, the following improvements can be considered by future developers:

- Dynamic domain switching via filenames (different SCXML files). Each SCXML file considered a different FSM for a different domain. Construct rules for task domain selection and additional rules for out-of-domain utterance detection, that involves the change of domain, if such makes sense for the dialogue context;
- Integration with a complete robotic platform and testing with multiple robots;
- Voice differentiation. Differentiate a female voice from a male voice or an elder voice from a child voice;
- Use of databases to register user information and dialogue history, such as number of times a user interacted with the system and define a "skill level" to change between interaction modes;
- Restrict the set of recognizable words or phrases that can be expected to reduce the risk of speech recognition mistakes;
- Generate one recognition grammar file from another (ASR – NLU or vice-versa) in order to reduce duplicated information;
- Improve the interaction by adding different options before the main dialogues take place: give the option to choose the gender of the voice and also, if available, an adult or elder voice; ask the user what kind of interaction is desired: for instance, with more explanations/suggestions than normal and where the sentences are spoken in a slower rate.

Bibliography

[1] A. Teixeira, D. Braga, L. Coelho, J. A. Fonseca, J. Alvarelhão, I. Martín, A. Queirós, N. Rocha, A. Calado, and M. Dias, "Speech as the basic interface for assistive technology," in *Proc. Int. Conf. on Software Development for Enhancing Accessibility and Fighting Info-Exclusion*, 2009.

[2] J. Jiang, F. Khelifi, P. Trundle, and A. Geven, "HERMES: a FP7 funded project towards the development of a computer-aided memory management system via intelligent computations," *J. Assist. Technol.*, vol. 3, pp. 27–35, 2009.

[3] "CARL (Communication, Action, Reasoning and Learning in Robotics)," 2003. [Online]. Available: http://www.ieeta.pt/carl/. [Accessed: 14-Jun-2014].

[4] R. A. M. Braga, M. Petry, L. P. Reis, and A. P. Moreira, "IntellWheels: Modular development platform for intelligent wheelchairs," *J. Rehabil. Res. Dev.*, vol. 48, pp. 1061–1076, 2011.

[5] "IntellWheels | A Cadeira de Rodas Inteligente que soma prémios." [Online]. Available: http://www.pofc.qren.pt/Media/Noticias/entity/IntellWheels--A-Cadeira-de-Rodas-Inteligente-que-soma-premios. [Accessed: 14-Jun-2014].

[6] B. M. Faria and L. P. Reis, "User Modeling and Command Language Adapted for Driving an Intelligent Wheelchair," in *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2014, no. Ii, pp. 158–163.

[7] P. A. H. Polónio, "Cambada @ home," Masters' Thesis in Computers and Telematics Engineering - Department of Electronics, Telecommunications and Informatics from University of Aveiro, 2010.

[8] "Living Usability Lab." [Online]. Available: http://www.microsoft.com/pt-pt/mldc/lul/pt/pt/default.aspx. [Accessed: 14-Jun-2014].

[9] J. Cunha, J. L. Azevedo, M. B. Cunha, L. Ferreira, P. Fonseca, N. Lau, C. Martins, A. J. R. Neves, E. Pedrosa, A. Pereira, L. Santos, and A. J. S. Teixeira, "CAMBADA@Home'2013: Team Description Paper." 2013.

[10] R. S. Nickerson, "On conversational interaction with computers," in *Proceedings of the ACM/SIGGRAPH workshop on User-oriented design of interactive graphics systems - UODICS '76*, 1977, p. 101.

[11] F. M. G. Quinderé, "Comunicação Humano-Robô através de Linguagem Falada," Phd Thesis in Electrotecnic Engineering - Department of Electronics, Telecommunications and Informatics from University of Aveiro, 2013.

[12] M. McTear, *Spoken Dialogue Technology: Towards the Conversational User Interface*. Springer Science & Business Media, 2004.

[13] "Apple - iOS 7 - Siri." [Online]. Available: http://www.apple.com/ios/siri/. [Accessed: 16-May-2014].

[14] "Windows Phone 8.1 and Cortana | Windows Phone (United States)." [Online]. Available: http://www.windowsphone.com/en-us/features-8-1. [Accessed: 19-May-2014].

[15] A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi, "Let's Go Public! Taking a Spoken Dialog System to the Real World," *INTERSPEECH*, 2005.

[16] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen J., and L. Hetherington, “Jupiter: A Telephone-Based Conversational Interface for Weather Information,” *IEEE Trans. Speech Audio Process.*, pp. 100–112, 2000.

[17] O. Lemon, A. Bracy, A. Gruenstein, and S. Peters, “The WITAS multi-modal dialogue system I,” *Eurospeech - Scand.*, pp. 4–7, 2001.

[18] D. Jurafsky and J. H. Martin, “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition,” 2nd ed., Upper Saddle River, NJ, USA: Pearson, Pretice-Hall, 2008.

[19] “ASIMO by Honda | The World’s Most Advanced Humanoid Robot.” [Online]. Available: http://asimo.honda.com/. [Accessed: 23-May-2014].

[20] “Aldebaran Robotics | Humanoid robotics & programmable robots.” [Online]. Available: http://www.aldebaran.com/en. [Accessed: 23-May-2014].

[21] “ASK NAO, Autism Solution for Kids.” [Online]. Available: http://asknao.aldebaran-robotics.com/. [Accessed: 23-May-2014].

[22] PAL Robotics, “REEM-C: Robotics Research,” 2014. [Online]. Available: http://pal-robotics.com/en/products/reem-c/. [Accessed: 15-Jun-2014].

[23] C. Theobalt, J. Bos, T. Chapman, A. Espinosa-Romero, M. Fraser, G. Hayes, E. Klein, T. Oka, and R. Reeve, “Talking to Godot: dialogue with a mobile robot,” *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 2, no. 1, pp. 1338–1343, 2002.

[24] I. Kruijff-Korbayová, G. Athanasopoulos, A. Beck, P. Cosi, H. Cuayáhuitl, T. Dekens, V. Enescu, A. Hiolle, B. Kiefer, H. Sahli, M. Schröder, G. Sommavilla, F. Tesser, and W. Verhelst, “An Event-Based Conversational System for the Nao Robot,” in *Proceedings of the Paralinguistic Information and its Integration in Spoken Dialogue Systems Workshop*, 2011, pp. 125–132.

[25] N. Mitsunaga, T. Miyashita, H. Ishiguro, K. Kogure, and N. Hagita, “Robovie-IV: A Communication Robot Interacting with People Daily in an Office,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5066–5072.

[26] H. Ishiguro, T. Ono, M. Imai, T. Maeda, T. Kanda, and R. Najatsu, “Robovie: An Interactive Humanoid Robot,” *Ind. Robot An Int. J.*, vol. 28, no. 6, pp. 498 – 504, 2001.

[27] K. Kaneko, K. Harada, F. Kanehiro, G. Miyamori, and K. Akachi, “Humanoid robot HRP-3,” *2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, Sep. 2008.

[28] F. Hegel, T. Spexard, B. Wrede, G. Horstmann, and T. Vogt, “Playing a different imitation game: Interaction with an Empathic Android Robot,” in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, 2006, pp. 56–61.

[29] R. Tellez, F. Ferro, S. Garcia, E. Gomez, E. Jorge, D. Mora, D. Pinyol, J. Oliver, O. Torres, J. Velazquez, and D. Faconti, “Reem-B: An autonomous lightweight human-size humanoid robot,” in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, 2008, pp. 462–468.

[30] L. P. Reis, R. A. M. Braga, M. Sousa, and A. P. Moreira, “IntellWheels MMI: a flexible interface for an intelligent wheelchair,” in *RoboCup 2009: Robot Soccer World Cup XIII*, Springer-Verlag, 2010, pp. 296–307.

[31] R. Bischoff and V. Graefe, "HERMES—an Intelligent Humanoid Robot Designed and Tested for Dependability," in *Experimental Robotics VIII*, 2003, pp. 64–74.

[32] V. Graefe and R. Bischoff, "Three examples of learning robots," *Proc. Int. Conf. Control. Autom. Syst.*, 2001.

[33] R. Bischoff and T. Jain, "Natural communication and interaction with humanoid robots," *Second Int. Symp. Humanoid Robot.*, no. October, 1999.

[34] N. Mavridis, M. Petychakis, A. Tsamakos, P. Toulis, S. Emami, W. Kazmi, C. Datta, C. BenAbdelkader, and A. Tanoto, "FaceBots: Steps towards enhanced long-term human-robot interaction by utilizing and publishing online social information," *Paladyn*, vol. 1, no. 3, pp. 169–178, Feb. 2010.

[35] J.-D. Dessimoz, peter F. Dominey, M. R. Elara, D. Gossow, D. Holz, L. Iocchi, G. Kraetzschmar, F. Mahmoudi, D. Nardi, S. Olufs, C. Rascon, J. Ruiz-del-Solar, P. E. Rybski, J. Savage, S. Schiffer, J. Stückler, K. Sugiura, T. van der Zant, S. Wachsmuth, T. Wisspeintner, J. Xie, and A. Yazdani, "RoboCup @ Home Rules & Regulations." 2013.

[36] X. Chen, D. Lu, K. Chen, Y. Chen, and N. Wang, "KeJia: The Intelligent Service Robot for RoboCup@ Home 2014." 2014.

[37] L. Ziegler, J. Wittrowski, S. Meyer, and S. Wachsmuth, "ToBI - Team of Bielefeld : The Human-Robot Interaction System for RoboCup @ Home 2014." 2014.

[38] L. Pineda, C. Rascon, G. Fuentes, V. Estrada, A. Rodriguez, I. Meza, H. Ortega, M. Reyes, M. Peña, J. Duran, E. Campos, S. Chimal, and A. Orozco, "The Golem Team, RoboCup@ Home 2014," *turing.iimas.unam.mx*. 2014.

[39] M. Bruinink, M. Rudinac, F. Gaisser, A. C. A. B, G. Liqui, M. Wisse, and P. Jonker, "Delft Robotics RoboCup @ Home 2014 Team Description Paper." 2014.

[40] G. Détrez, R. Enache, B. Bringert, H. Burden, H.-J. Daniels, M. Forsberg, K. Johannisson, J. Khegai, P. Ljunglöf, and P. Mäenpää, "Grammatical Framework: a programming language for multilingual grammar applications." [Online]. Available: http://www.grammaticalframework.org/. [Accessed: 23-May-2014].

[41] M. Nakano, Y. Hasegawa, K. Nakadai, T. Nakamura, J. Takeuchi, T. Torii, H. Tsujino, N. Kanda, and H. G. Okuno, "A two-layer model for behavior and dialogue planning in conversational service robots," *2005 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 3329–3335, 2005.

[42] T. Belpaeme, P. E. Baxter, R. Read, R. Wood, H. Cuayáhuitl, B. Kiefer, S. Racioppa, I. Kruijff-Korbayová, G. Athanasopoulos, V. Enescu, R. Looije, M. Neerincx, Y. Demiris, R. Ros-Espinoza, A. Beck, L. Cañamero, A. Hiolle, M. Lewis, I. Baroni, M. Nalin, P. Cosi, G. Paci, F. Tesser, G. Sommavilla, and R. Humbert, "Multimodal Child-Robot Interaction: Building Social Bonds," *J. Human-Robot Interact.*, vol. 1, no. 2, pp. 33–53, Jan. 2013.

[43] J. Baldridge and G. G.-J. M. Kruijff, "Multi-modal combinatory categorial grammar," in *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - EACL '03*, 2003, vol. 1, p. 211.

[44] H. Asoh, Y. Motomura, F. Asano, I. Hara, S. Hayamizu, K. Itou, T. Kurita, T. Matsui, N. Vlassis, R. Bunschoten, and B. Krose, "Jijo-2: an office robot that communicates and learns," *IEEE Intell. Syst.*, vol. 16, no. 5, pp. 46–55, Sep. 2001.

[45] D. Regateiro, “External Spoken Dialog System: A client/server Implementation for the CAMBADA@Home robot,” Technical Report, IEETA, 2012.

[46] Open Source Robotics Foundation, “ROS.org | Powering the world’s robots.” [Online]. Available: http://www.ros.org/. [Accessed: 25-Jun-2014].

[47] MIT, “Java-WebSocket: A barebones WebSocket client and server implementation written in 100% Java,” 2013. [Online]. Available: http://java-websocket.org/. [Accessed: 18-Jun-2014].

[48] Kerryjiang, “SuperWebSocket, a .NET WebSocket Server - Home,” 2013. [Online]. Available: http://superwebsocket.codeplex.com/. [Accessed: 17-Jun-2014].

[49] Microsoft, “Microsoft Speech Platform SDK 11 Documentation,” 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd266409(v=office.14).aspx. [Accessed: 17-Jun-2014].

[50] S. Andrew Hunt and H.-P. Scott McGlashan, “Speech Recognition Grammar Specification Version 1.0,” 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-speech-grammar-20040316/. [Accessed: 06-Jun-2014].

[51] W. Ward and B. Pellom, “The Phoenix Parser User Manual.” 2002.

[52] W. Ward, S. Issar, X. Huang, H.-W. Hon, M.-Y. Hwang, S. Young, M. Matessa, F.-H. Liu, and R. Stern, “Speech understanding in open tasks,” in *Proceedings of the workshop on Speech and Natural Language - HLT ’91*, 1992, p. 78.

[53] W3C, “State Chart XML (SCXML): State Machine Notation for Control Abstraction,” 2010. [Online]. Available: http://www.w3.org/TR/2014/WD-scxml-20140529/. [Accessed: 18-Jun-2014].

[54] W3C, “Voice Extensible Markup Language (VoiceXML) 3.0,” 2010. [Online]. Available: http://www.w3.org/TR/voicexml30/. [Accessed: 18-Jun-2014].

[55] W3C, “Speech Synthesis Markup Language (SSML) Version 1.0,” 2004. [Online]. Available: http://www.w3.org/TR/speech-synthesis/. [Accessed: 18-Jun-2014].

[56] Microsoft, “Emphasis Element,” 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/hh361617(v=office.14).aspx. [Accessed: 05-Jun-2014].

[57] K. Dautenhahn, “The AuRoRA Project.” [Online]. Available: http://www.aurora-project.com/. [Accessed: 20-Jun-2014].

[58] K. Dautenhahn, “Socially intelligent robots: dimensions of human-robot interaction.,” *Philos. Trans. R. Soc. Lond. B. Biol. Sci.*, vol. 362, no. 1480, pp. 679–704, Apr. 2007.

[59] J. Brooke, “System Usability Scale.” 1986.

[60] J. Brooke, “System Usability Scale (SUS),” 06-Sep-2014. [Online]. Available: http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. [Accessed: 02-Jul-2014].